NASA Technical Memorandum 104751

Making Intelligent Systems Team Players: Overview for Designers

Jane T. Malin
Lyndon B. Johnson Space Center
Houston, Texas

Debra L. Schreckenghost
MITRE
Houston, Texas

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas

June 1992

Abstract

This report is a guide and companion to the NASA Technical Memorandum 104738, "Making Intelligent Systems Team Players", Vols. 1 and 2. The first two volumes of this Technical Memorandum provide comprehensive guidance to designers of intelligent systems for real-time fault management of space systems, with the objective of achieving more effective human interaction with these systems. This report provides an analysis of the material discussed in the Technical Memorandum. It clarifies what it means for an intelligent system to be a team player, and how such systems are designed. It identifies significant intelligent system design problems and their impacts on reliability and usability. Recommendations are made for situations in which common design practice is not effective in solving these problems. The main points in the Technical Memorandum are summarized, with references to that document for further information.

Keywords: human-computer interaction, intelligent systems, real-time space operations, design guidance, information requirements, user interface

Table of Contents

List of Figures

List of Tables

Section 1
Introduction

1.1 Purpose and Objectives of Report

Space operations support provides a challenging domain for applications of advanced automation technology. The increased availability of workstations equipped with such technology has resulted in a wave of intelligent system prototype applications in space operations. Unfortunately, the prototypes do not always live up to the bright expectations the technology engenders. Common criticisms are that intelligent systems have badly designed user interfaces and are hard to understand, resulting in user distrust or disregard.

A case study of NASA intelligent system (IS) prototypes for real-time fault management of space systems has led to the conclusion that the design problems are not located just at the user interface, but are also in the design of the IS. Intelligent system designers need help to achieve more effective human interaction. The NASA Technical Memorandum 104739, Making Intelligent Systems Team Players: Human-Computer Interaction Design (Malin *et al.*, 1991) was published to provide guidance in designing intelligent systems (ISs). The goal is intelligent systems that effectively support human operators (flight controllers and crew) in performing their tasks, making these systems a part of the space operations support team. The Tech Memo[1] is lengthy and detailed. We have written this report as a guide and companion to the Tech Memo. It serves two purposes, which are described below.

First, this report provides an analysis of the material discussed in the Tech Memo. It clarifies what it means for intelligent systems to be team players, and how such systems are designed. It identifies the significant intelligent system design problems and their impacts to reliability and usability, and makes recommendations when common design practice is not effective in solving these problems. It summarizes and distills this large body of material into a concise description of the important, new concepts concerning human-computer interaction (HCI) design for the intelligent systems being built today. It also points to needed research in artificial intelligence (AI) and human-computer interaction that will assist designers building the intelligent systems of tomorrow.

Second, this report provides an overview (or "road map") of the material discussed in the Tech Memo. As a road map, it identifies the important characteristics of the terrain (i.e., summarizes the main points in the Tech Memo). It also provides directions to specific destinations (i.e., provides indices into sections of the Tech Memo). As a written document, the material in the Tech Memo necessarily has a single, fixed organization. Yet, there are multiple types of "users" of the Tech Memo (e.g., IS designers, researchers) with different perspectives on how to use the Tech Memo. We have included a section describing the material in the Tech Memo from the perspective of the intelligent system designer (see section 3).

1.2 Scope and Organization of Report

This report was written for use primarily by intelligent system designers as a source of design guidance, and for HCI and AI researchers as a source of application challenges requiring additional research. It should also be of interest to users defining requirements for intelligent software, software engineers interested in incorporating HCI design into intelligent system design, and managers of intelligent system development projects.

---

[1] The Memorandum (Malin *et al.*, 1991) will be referred to as "the Tech Memo" in the remainder of this report.

Section 1 introduces our purpose in writing this report, provides relevant background material, and describes how to use this report. Our important, new concepts are analyzed and discussed in section 2. Section 3 is written specifically for the intelligent system designer. It is organized around the major design and analysis tasks, resembling a guide or instruction manual for intelligent system design.

1.3 Background: Space Fault Management Tasks and Terms

This section provides a brief description of space fault management, including a summary of tasks and definitions of terminology, for readers not familiar with this domain. Readers who are familiar with space operations may skip this section.

Ground-based space operations consist primarily of monitoring and control operations. An important aspect of these operations is managing faults in the domain system being monitored and controlled (what we call the *monitored process*). There are two primary goals of fault management: (1) to maintain safe operations during anomalies, and (2) to minimize the impact of anomalies to mission objectives. Safe operations are operations that do not pose a threat to either the crew or the vehicle. These primary goals may be partitioned into four sub-goals: (1) to detect and compensate for anomalies, (2) to identify the cause of anomalies when possible, (3) to recover the faulty capability when possible, and (4) to maintain safety and mission objectives consistent with the remaining functional capability when recovery is not possible. There are additional constraints on fault management operations. These goals must be achieved with a minimum of disturbance to ongoing operations unaffected by the anomalous behavior, and must not compromise baseline safety and capability criteria as outlined in the flight rules. The major fault management tasks performed to accomplish these goals are listed below:

- *behavior monitoring*: monitoring the behavior of both the monitored process and the intelligent system.
- *anomaly detection*: discovering when anomalous behavior is exhibited by either the monitored process or intelligent system.
- *anomaly compensation*: mitigating the effect of an anomaly; compensation can be temporary while fault diagnosis proceeds, or permanent if fault recovery is not possible (safing[2] and reconfiguration[3] are common ways to compensate for anomalies).
- *mission impact assessment*: evaluating how anomalies, and the procedures executed to manage anomalies, affect mission goals.
- *anomaly propagation*: predicting the effects of an anomaly over time.
- *functional capability assessment*: determining the functional capability affected by an anomaly, and the functionality remaining to accomplish mission goals.
- *fault diagnosis and testing*: determining the possible causes of an anomaly, and testing to minimize this set of possible causes.
- *fault identification*: uniquely determining the cause of an anomaly.

---

[2] safing: process of identifying the safety impacts resulting from an event and taking appropriate action to put the crew and vehicle into a safe configuration

[3] reconfiguration: process of altering the configuration of the affected system to minimize safety and mission impacts

- *fault recovery or repair*: modifying the faulty item to correct the fault and repair anomalous behavior.
- *modification of mission plans or objectives*: altering mission plans or goals in response to the impacts of an anomaly.

A few definitions are required to understand these tasks. A *mission* corresponds to a set of tasks associated with a specific set of objectives (e.g., deploy a payload). *Procedures* are elements of a plan that specify the sequences of activities performed by the space operations support team to achieve safety and mission objectives within flight constraints. *Activities* are portions of tasks that have been assigned to specific agents, and that employ available resources (e.g., information, capabilities) to achieve the goals supported by these actions. *Anomalies* are irregular system behaviors or conditions that can be caused by a variety of influences, including faults, environmental factors, operator error, etc. A *fault* is the cause of a loss of functionality in a system, subsystem, component, or part. For example, one anomaly of a Display and Control panel is that a light did not glow when expected. The associated fault is a burnt-out light bulb.

This section has given a very brief overview of fault management operations for space systems. Section 3 in the Tech Memo (Malin *et al.*, 1991) contains a more thorough discussion of this topic.

Section 2
Intelligent System Design Concepts

The theme of the Tech Memo (and even its title) is "making intelligent systems team players". This phrase implies a lot about the design of intelligent systems and their user interfaces. In this section we summarize the important concepts in the Tech Memo, highlighting what is new and different about our approach to designing intelligent systems for human interaction. We discuss characteristics of space operations that constrain intelligent system design and lead to the team player concept. We analyze what it means for an intelligent system to be a team player, and how to prevent common information problems that interfere with teamwork. We discuss how to design reliable and usable intelligent systems, with a focus on defining information requirements.

2.1 Space Operations Constraints on Intelligent System Design

Technology advances have enabled the use of intelligent systems during real-time space operations. Intelligent systems used for space operations are a type of complex interactive real-time software system. In common with such complex software systems, intelligent system operations are characterized by (1) large amounts of information provided continuously from multiple sources, (2) time-constrained processing with deficiencies in information quality and availability, (3) sophisticated software capability and sophisticated users, often with multiple tasks being performed concurrently or jointly by human and computer, and (4) the potential for active information exchange and interaction between human and computer. Implicit in this characterization of the space operations environment are assumptions about the domain and how the intelligent system operates within the domain. In this section, we discuss the characteristics of space operations that affect intelligent systems design today, and what they imply about IS design needs in the future.

Space operations are continuous and the situation changes frequently. Space systems are very dynamic and complex. Even under nominal conditions, situations can change quickly, requiring the space operations support team to continuously monitor the systems and respond to changes in real-time. When anomalies do occur, space operations cannot be stopped while the problem is being fixed. The space operations support team must correct the affected portions of the system while maintaining operational capability in the unaffected portions of the system. As a result, the space operations support team must be able to perform multi-tasking and complex sequences of events and operations. The support team must also be able to respond to unexpected situations with novel operations. To accommodate these conditions, the IS must execute in real time. The IS design should include some type performance control (e.g., for the IS being built today, such capability includes disabling portions of a rule base, or at least bypassing some data sets to "catch up" to real time). The intelligent system should also be designed for unexpected situations, by providing alternate ways of performing operations that provide the human operator some flexibility in responding to such situations. In the long run, the human operator should be able to "inform" the IS of modified operations and to repair the IS when it is not performing well.

Space operations involve risk. Errors in performing space operations can risk the safety of crew and vehicle, and the ability to accomplish mission objectives. When introducing a new technology into space operations, it is important to understand the potential of that technology to increase these risks (i.e., the potential of introducing hazards that could lead to accidents caused by the support system; Leveson, 1991). Intelligent systems can introduce risk into critical operations directly by taking erroneous action, or indirectly by causing operator error. To avoid such risks, a variety of precautions can be taken. For example, a common policy aimed at reducing such risks is to prohibit commanding of the

vehicle by an IS. Also, the human operator is typically responsible for supervising the IS, and assessing if its recommended actions are risky. A less obvious but important approach is to present information from the IS in such a way that it doesn't overload or distract the operator (e.g., minimize unimportant interruptions of operator tasks). At a minimum, intelligent systems should be designed to avoid increasing risk. Intelligent systems that reduce the risk in space operations are a longer term goal requiring additional research.

Measurements will be noisy, unreliable, and will periodically drop out. Measurements from space-based systems are frequently degraded (e.g., noisy or intermittently flawed) or unavailable (e.g., short duration data dropout, loss of signal while transmitting satellite unavailable). Yet real-time space operations require continuous, online support. The intelligent system will be required to operate using such measurements without behaving anomalously. For example, it should not be necessary to shut down the IS at loss of data, or to ignore false alarms caused by noisy input. Measurement preprocessing to remove bad data prior to IS processing is a common approach to minimizing the effect of bad measurements. The ability for the intelligent system to actively compensate for loss of information, and to effectively accomplish tasks using partial information, is an important research issue.

Large quantities of information must be assimilated quickly and accurately by the human operator. Dynamic, complex space systems generate large quantities of rapidly changing data. Operating such systems requires considerable design and domain knowledge as well. If not managed, this flood of information can confuse and overload the human operator. Since the intelligent system generates yet more information, it should be designed to assist the human operator in effectively using its information in conjunction with other information (e.g., IS conclusions should be presented in conjunction with related data, instead of making the operator search to find related data). Intelligent systems in use today should avoid increasing the information load on operators. For example, providing explanatory information using a retrospective, conversational interaction style can distract the operator, causing the operator to miss important changes in the current situation. In future systems, the intelligent system may play a more active role, by integrating and managing diverse information to reduce operator workload.

Human operators often misunderstand what the intelligent system is capable of doing. Like many new technologies, intelligent systems are often poorly understood by the human operators using them. User expectations of technology, often based on state-of-the-art capabilities, can differ significantly from the operational realization of that technology. Intelligent systems are often perceived as "magic" (Abbott, 1991) and attributed with performing tasks significantly different from their actual tasks (and often well outside the scope of current capability). To avoid the perception of a "magical" system, the human operator should find it easy to continuously understand what the IS is doing and why it is doing this (i.e., its goals and the activities performed to achieve them). With such an understanding the human can assess if IS conclusions are correct, and is better able to respond to IS errors. For intelligent systems being built today, this means providing the operator with information about the tasks and current line of reasoning of the IS. For intelligent systems of the future, research is needed on building systems whose reasoning and tasks are evident from their representation and presentation of information about situations as they develop.

The intelligent system will behave anomalously at some time during operational use. Space operations consist of difficult, complex activities. Situations can occur that have not been encountered before, or even anticipated during design. In such an environment, we can safely assume that the intelligent system will eventually make errors. When the IS behaves anomalously, it must either be repaired in real time, or its tasks must be performed in

another way. Self repair is possible for some types of IS errors. For example, a self-repairable IS can correct itself when predictable errors occur (e.g., retract conclusions based on bad data). Not all IS errors can be so easily repaired, however. An approach suited to the capability of today's intelligent systems is compensating for irreparable IS errors by providing alternate modes of operation independent of the IS. For example, when the intelligent system fails, the human operator should be able to access information and functionality needed to perform tasks normally assigned to the IS. Since repairable intelligent systems are more complex to build and such capability is limited in the IS being built today, this represents an important application challenge for the AI research community.

Human operators in space operations are experts in their domain. A common assumption in the development of an intelligent system is that the system will be more knowledgeable than its user. This assumption does not hold for intelligent systems used in space operations, since the typical user is a flight controller who is also a domain expert. Intelligent systems for space operations should support and assist their human supervisors in performing tasks, not tutor them about the domain. For intelligent systems being built today, this means designing for situations where the IS gives the wrong advice. In the simplest case, the human operator should be able to ignore the IS advice. A more sophisticated approach is to permit the human to correct the IS (i.e., the intelligent system can receive advice as well as give it; Martin and Firby, 1991). Future intelligent systems may even be able to "learn" from their human supervisors.

The intelligent system will often be embedded in a larger support system. Intelligent systems for space operations are usually part of an integrated support system that includes conventional and display software. The intelligent system should not interfere with the use of the non-AI software, even when the IS fails. The human operator should be able to identify the source of the displayed information. When a component of the support system fails, such as the IS, the human operator should be informed of what information has been lost and what is unaffected. Alternate ways of accomplishing mission objectives without the lost information should be also be considered during system design and training. For example, when the intelligent system fails, the operator should be informed of what information is no longer available, and should be able to continue viewing information from the remaining support software. At a minimum, the IS should not interfere with the human operator's tasks. The integration of diverse information for support of the human operator's tasks is a longer term research issue related to this.

2.2 The Intelligent System as a Team Player

Real-time space operations are typically performed by teams of humans. These teams have a common set of goals (safety and mission objectives) that are accomplished by team members performing tasks both individually and cooperatively. When multiple team members are assigned to cooperatively perform tasks, they must coordinate their individual activities and collaborate to exchange necessary information. Team members may have overlapping areas of expertise (i.e, specialties), and have some capability to back up each other's specialties.

We have extended this team concept to groups of humans and intelligent systems. The human and intelligent system are considered as members of a team that manages faults in the domain system being monitored (the monitored process). What does it mean for an intelligent system to be a team player in the space operations environment?

- A team player is reliable
- A team player communicates effectively with others

- A team player coordinates individual activities with others
- A team player is guided by the coach

In this section, we describe these traits of a team player, based on the way intelligent systems are used in space operations (see section 2.1 for a description of the space operations constraints on intelligent system design).

A Team Player is Reliable

Space operations are characterized by uncertainty (e.g., noisy data), complexity, and risk. In such an environment, a reliable team member is one who will try to do the right thing, and, in the worst case, will not cause harmful effects or get in the way. Such a team member should reliably perform assigned tasks, which requires being capable of alternate ways of doing things. For an intelligent system, this means designing a system that is both robust and flexible. Verification and validation techniques are used to determine if the system is "doing the right thing" in expected situations. But what if the unexpected occurs? Robust, flexible intelligent systems will include some provision for continuing operations in situations where things don't go as expected. Such situations include the IS behaving anomalously (e.g., drawing a wrong conclusion, pursuing an unproductive path of investigation), or the human operator altering nominal or accepted procedures (e.g., executing a contingency procedure unknown to the IS, changing task allocation due to workload problems or unacceptable performance).

When intelligent system behavior or performance is not as planned, a reliable IS will either support alternate ways of accomplishing its tasks or permit itself to be repaired for continued operations. To ensure that the IS doesn't obstruct operations, the human should at least be able to disable the IS and perform all its tasks (i.e., reallocate IS tasks). If the IS is embedded in a larger support system, it must permit access to information and capability necessary for a human to do IS tasks when the IS is disabled, and should clearly identify the disabled state of the IS on the display. For example, when one IS from the case study is disabled, it displays the portions of the screen used by the IS in a grey shade. Task reallocation can be graded, from a partial reallocation (e.g., disable a portion of a rule base) to a complete override of all task responsibilities (e.g., turn off the IS). Providing for task reallocation is the most likely approach to building reliable intelligent systems today. There are thorny design problems associated with task reallocation, however, such as how a disabled IS can effectively resume its responsibilities at a later time (i.e., what information is needed to bring the IS back "up to speed" after it has been stopped).

The second approach to building reliable intelligent systems is to provide the ability for the IS to be repaired. The repair can either be performed by the human or the IS (i.e., self repair). Repairing intelligent system behavior consists of modifying or correcting its activities, reasoning, or the information used by it. To repair the IS, the human must also be able to understand the IS activities leading up to the error (why the error occurred) and to monitor the effects of the repair (did the IS recover from its error; cf., section on Coach). Although real-time repair of intelligent systems is beyond the capability of most systems today, a limited form of IS repair can be achieved by designing for the human to provide or correct data used by the IS (e.g., correct noisy data value causing the IS to make wrong conclusions). It is also possible for the IS to perform some type of self repair (e.g., designate uncertain conclusions provisionally true, retract conclusions based on bad data). Further in the future is the ability for the human to alter the knowledge and reasoning of the IS in real time, permitting a form of on-the-job training of the IS by the human.

A Team Player Communicates Effectively with Others

Effective human-computer communication requires striking a balance. On one hand, the intelligent system must provide enough information for the human to understand what it is doing and why (i.e., enough information to prevent the IS being perceived as "magic"). On the other hand, the intelligent system must not provide so much information that it interrupts or distracts the human from more important tasks. The human operator is already overloaded with information. The problem of information overload becomes especially important in real-time environments where complex, high risk tasks are performed, and where human errors can represent serious risk. Essential to achieving this balance when designing the IS is identifying when it is important to communicate, and when to keep quiet. Additionally, the IS should be clear about what it is communicating (i.e., don't bury the important information in a pile of unimportant data).

Most of the intelligent systems in the case study communicate with the human in one or more of the following ways:

- *message list*: a chronological list of state and status assessments and/or action recommendations
- *annotated schematic*: a graphic representation of the physical structure of the system, annotated with sensor measurements or state/status assessments from the IS
- *explanation*: a conversational style of providing additional justification for an IS conclusion

All of these methods have significant limitations in achieving effective communication in this environment.

Both message lists and schematics can obscure intelligent system information important to the human operator (Woods *et al.*, 1991). Situations develop as patterns of events indicated by changing state and status. Schematics can only present the latest event (i.e., the current state). Additionally, if events are not related to physical structure (e.g., functional status), they can be difficult to present clearly using a schematic. Message lists do capture some event history, but do not represent the relationships between events (e.g., events occurring in parallel, the temporal distance between events) necessary to reveal these patterns. Since chronology is the means of sorting messages, related information can become dissociated.

Intelligent system designers should consider alternatives to message lists and schematics that assist the human operator in seeing patterns of events as they occur. For example, Woods and Potter are investigating timelines as an alternative to message lists (Potter and Woods, 1991). Woods (Woods *et al.*, 1991) recommends considering alternatives to the physical schematics as well. If schematics are deemed appropriate, they recommend approaches for emphasizing the information important to the task. Woods is also investigating representation aiding to identify the important changes in situation and to focus the operator's attention on those changes (Woods, 1991). See section 5 of the Tech Memo for recommendations concerning message lists and schematics.

The third common approach to communication is explanation. Most explanation systems operate retrospectively, requiring the operator to wait until after a situation has stabilized (and the IS has reached a conclusion) before attempting to describe what happened. In real-time support environments, the operator cannot afford to wait until system behavior stabilizes, for the safety impacts may be too great. Such event reconstruction is also not sufficient for team coordination during joint activities (cf., section on Coordination). Additionally, the conversational style of explanation can be distracting and can contribute to information overload.

Even assuming that the human operator has time to hold a conversation with the IS, a problem remains with traditional approaches to explanation. Affecting the behavior of a human operator requires that the human both *understand* the meaning and consequences of the explanation, and *accept* them as correct. Most explanation systems assume that failure to influence human behavior occurs because the human failed to understand the explanation, and continue to provide more detailed justification directed at improving understanding. But, contrary to this assumption, acceptance does not necessarily result from understanding. The human may understand the intended meaning but choose not to believe it, due to information unknown to the IS or not considered by the IS. Or the human may believe the information but be unwilling to alter behavior, due to the belief that adverse side-effects will result from altering behavior or that the consequences of altering behavior are of no significance. This type of explanation ignores that expert users (such as flight controllers) usually know MORE about the domain than the IS, and that the purpose of the explanation is often to clarify what the IS "thinks" is happening so the user can detect IS anomalies.

For the intelligent systems being designed today, the use of such retrospective explanation approaches for real-time tasks should be limited. When it is necessary to provide an alternative to explanation, it must be custom designed. For example, Martin and Firby provide an explicit representation of robot tasks (both planned and ongoing tasks) to the human monitoring the robot (Martin and Firby, 1991). The human can use this task representation as a means of understanding the robot's strategy for accomplishing goals and altering those strategies by modifying the planned tasks.

Thus, providing alternatives to explanation is an important research area for real-time intelligent systems. These alternatives should address both the problem of retrospective presentation and the deeper problem of clarifying IS reasoning. A promising area for investigation is using *shared representations* as a means of human-intelligent system communication. A representation shared between agents is prevalent as the basis of communication in many domains, including humans advising robots using a shared task representation (Martin and Firby, 1991), distributed machine agents planning tasks using shared goals (Decker *et al.*, 1991), and designers developing shared mental models (Sycara and Lewis, 1991). Using a shared representation, the human should be able to follow operational situations and develop assessments consistent with IS assessments (i.e., a shared view) as part of the normal monitoring and control operations. Additionally, this approach permits developing common knowledge between the human operator and intelligent system, requiring the operator to expend less attention (and time) to effectively supervise the IS (cf. section on Coach). The long term goal of such efforts is that intelligent system conclusions should be represented and presented in such a way as to be self-evident to the human operator; in effect, such an intelligent system becomes *self-explanatory*.

An important task in designing for communication via shared representations is identifying the information needed to develop such a shared view. Developing a shared view requires showing how a situation develops, including the important events that characterize the situation. These events include not only failures, but important state transitions and configuration changes in the monitored process. To share the IS view of a situation, the operator must also understand the bases of IS conclusions about that situation. We have seen intelligent systems that clarify their conclusions by presenting plots or tables of data

supporting these conclusions[1]. This is a simple but powerful example of how the human can share some portion of the world view of the intelligent system, so that both the human and the intelligent system are acting based on the same "understanding" of the situation. Providing such information results in an intelligent system that is used frequently, even during nominal operations. Such systems are "continuously useful", because users rely on them frequently. Such continuous use improves user trust and acceptance of these systems.

A Team Player Coordinates Individual Activities with Others

When teams of humans and intelligent systems operate in multi-tasking environments, it is necessary for them to coordinate their activities to avoid interfering with each other. A closer cooperation becomes important when team members jointly perform activities. Dependencies may exist between these activities that require frequent interaction and communication among team members, and that require timely reaction to changes in situation. When team members fail to coordinate their activities, team performance can be degraded. The effects range from reduced task performance to failure to achieve task objectives. Thus, degraded team performance can also negatively impact the monitored process.

These coordination tasks are performed in addition to fault management tasks and, to be a team player, the intelligent system must be designed to support them. For the systems being built today, this often means designing intelligent systems that won't interfere with the human performing fault management tasks. Team members should be able to back each other up (e.g., the human should be able to perform IS tasks as a contingency). Team members maintain awareness of each others activities, but focus their attention and efforts on the job to be done. To assist the operator in monitoring its activities, the IS should provide information to the operator about what it is doing (its activities), as well as what it has concluded. Such communication, however, should not overload, distract, or unnecessarily interrupt the operator's activities (cf., Communication). Team members don't operate in isolation, but are guided and coordinated by the coach. Since the human operator supervises the intelligent system members of the space operations support team (i.e., the human is the coach), the IS should permit some sort of human control over its activities (e.g., reassign tasks, correct bad data; cf., Coach).

In the future, shared human-intelligent system tasks will require that the human and the intelligent system be able to monitor each others activities and exchange information about the task (e.g, activities, goals, plans, beliefs, intentions). This information exchange should be based on an understanding common to the team of what is relevant to the current situation. Research is needed on ways to represent such information to achieve a common understanding of situation during real-time tasks (cf., Communication).

A Team Player is Guided by the Coach

Typically, intelligent systems are designed to advise humans, as off-line consultants or advisors to non-expert human problem-solvers. For complex space operations, however, the human operator is an expert and is more knowledgeable and dependable than the IS. Thus, the IS should be guided by the human. The human operator assumes a dual role in the space operations support team. The operator is both a team member who performs fault management activities, and a supervisor, or *coach*, who manages the activities and

[1] Two systems that provide data bases for IS conclusions are Vista, being developed by Rockwell (Horvitz *et al.*, 1992), and The Thermal Control System Advanced Automation Project, being developed by McDonnell Douglas (Hill, 1991).

performance of the team. As a team member, the human operator must be able to perform assigned tasks while maintaining expertise through on-the-job training. As the coach, the human operator must be able to monitor what the team is doing, direct individual members of the team, and coordinate their activities with respect to team goals. To do this, the human operator must understand what the IS is capable of doing, what it is doing at any given time (and why), and how its activities affect other members of the team. The operator must also be able to take some action when the either the team as a whole or an individual member of the team (such as the IS) is not performing as expected.

To guide an individual intelligent system, the human must be able to monitor and understand IS activities, and to detect and manage IS anomalies. Monitoring and understanding IS activities requires providing the human with a view into IS reasoning as a means of detecting anomalous IS behavior. The human should always understand what the IS is doing (its activities), and if its activities make sense and are having the expected effects. A representation shared between the human and the IS can make it easier for the human to understand and advise the intelligent system (cf., section on Communication). Intelligent system anomalies can be managed by either compensating for them or correcting them (cf., section on Reliability). To compensate for anomalous IS behavior, the human coach can reassign its tasks (task reallocation). To correct anomalous IS behavior, the human coach can alter information used by the IS, such as informing the IS of revised information when it has been misinformed. Task reallocation is the most common way of managing IS anomalies.

To supervise the team, the human must coordinate team activities in addition to managing anomalous behavior in the individual intelligent system (cf., section on Coordination). The human must monitor the effects of team activities to ensure they are as expected. The human must replan and coordinate team activities when these effects are not as expected or when goals are altered, or to compensate for an overworked or anomalous team member. At a minimum, the IS should not get in the way of efforts to coordinate the team (i.e., permit reassignment of its tasks, avoid interrupting unnecessarily). For a given fault situation there may be more than one way for the team to respond (e.g., correct an anomalous IS versus reassign its tasks). It is necessary for the human to perform a trade-off between these response options to determine what to do. Part of this trade-off includes assessing how each option will impact mission and safety goals, and evaluating the likelihood of unexpected or harmful outcomes associated with each option. Once action has been taken, the behavior of the team must be monitored to determine if the action was successful.

2.3 Information Design for Reliability and Usability

A key observation from the case study is that many perceived user interface problems in intelligent systems are actually information problems. These information problems result from (1) not providing the right information to support *domain tasks*, and (2) not recognizing that an intelligent system introduces *new user supervisory tasks* that require new types of information. These problems are made more difficult by failure to consider how intelligent systems operate in space environments (e.g., effect of data quality and availability on IS behavior; cf., section 2.1) and failure to integrate intelligent system operations with other operations. In this section we characterize these intelligent system design problems and recommend solutions to them. We describe how these design problems impact intelligent system reliability and usability, and delineate the benefits of solving them.

The first information problem is failure to provide the right information to support domain tasks. This problem results from not identifying the information that should be exchanged

to support the human operator's tasks (i.e., the information requirements). The most common tasks performed by intelligent systems being built today are fault monitoring and diagnosis. The information needed to support the operator in these tasks is the important behaviors, interesting relationships, and significant changes occurring in the monitored process (Woods *et al.*, 1991). To interpret this information, the operator must also understand the behavior expected to occur, and the thresholds delimiting significant or interesting changes (i.e., transition points). For example, many of the human operator's decisions during fault management require information about functional capability (what functionality has been lost, what is the mission impact of that loss). Yet the information typically communicated to the operator by the IS consists of component failures shown on schematics or listed in message logs. These methods of presenting information do not support identifying important changes (e.g., lost functionality) and relationships (e.g, how failures impact mission goals), and considerable effort is required to use this information to make fault management decisions. Thus, common practice in communicating with the human operator does not provide the information needed to make fault management decisions. To solve this design problem, information requirements should be defined based on a description of domain tasks. New user interface designs are also needed for effective presentation of information supporting functional assessments.

The second information problem is failure to design for user supervision. New user supervisory tasks include both monitoring ongoing IS activities with respect to mission goals and safety, and guiding and correcting the IS when it malfunctions. Intelligent systems are usually not designed to be managed because it is not well recognized that they need to be managed. This omission in design arises from two misconceptions: (1) that the intelligent system is more knowledgeable than its user (i.e., the human operator), and (2) that the intelligent system can be designed to prevent all errors from occurring. In fact, the typical space operations user (a flight controller) is also a domain expert. This expert user is more knowledgeable than the IS and is well qualified to supervise the IS. The misconception that all IS errors can be prevented results from unrealistic assumptions about the space operations environment and how intelligent systems operate within that environment. Due to the complexity of this environment, the behavior of the monitored process cannot always be accurately predicted, and unexpected situations occur. Because they are unexpected, the IS knowledge base does not address them and the IS can respond anomalously. Additionally, data problems (e.g., stale, noisy, biased) are common in space environments and can cause IS error. The intelligent system should be designed so that the user (the human operator) can take action to recover from such IS errors. In effect, the intelligent system can be designed to gracefully degrade outside the boundaries of its knowledge by providing the ability for the operator to correct or compensate for IS errors.

Typically, however, the intelligent system provides no means for the human operator to respond to IS errors, apart from turning off the IS. Thus, IS errors due to unanticipated anomalies or data deficiencies may require the operator to assume all IS responsibilities (i.e., take over from the IS). The IS must be designed for such dynamic reallocation of its tasks to the human operator. When tasks are reallocated from the IS to the operator, critical operational information and commanding capability used to manage the monitored process must be accessible independent from the IS. Note that for such task allocation, the human remains "in the loop" for all activities, contrary to the notion of the IS completely automating human tasks. Another way to design the IS for recovery from errors is to provide for repair or correction of the IS. This requires that the operator understand the capabilities and reasoning strategies of the IS well enough to manipulate them by altering information. It also requires that the IS be designed to accept such information changes.

Since part of managing the IS is minimizing the risks associated with supervisory activities, the IS should be designed to assist the operator in assessing these risks[2].

If not designed for user supervision, the intelligent system can also be difficult to understand (what we called a "magical" IS in section 2.1) because it doesn't provide the user with necessary information about IS processing. The user (the human operator) must understand what the IS can do (its capabilities), what it is currently doing (its activities), and why (its reasoning strategies). Without such an understanding, the human operator cannot hope to guide and correct the IS. Providing this additional information to supervise the IS can overload the operator, however, if it is not effectively managed. To manage information overload, it is important to identify what is relevant to the current task and emphasize that information on the user interface. Explanation is the traditional approach to clarifying and justifying IS conclusions. Explanation is not effective in real-time situations, however. Instead of retrospectively describing what occurred, it is necessary to provide a view of a situation as it develops, including the intelligent system's involvement in the situation. Additionally, explanation makes wrong assumptions about what information is needed by human operator. Instead of a tutorial, the expert user needs clarification of the IS reasoning strategies and the information used by the IS to come to conclusions, to supervise the IS by independently checking its conclusions.

A common design problem is failure to integrate intelligent system operations with other operations. Usually the intelligent system does not operate as a stand-alone system, but is instead embedded in a larger support system. The intelligent system must be integrated with the sources of operational data. Intelligent system integration can require significant non-AI software for managing data problems and displaying data. Data problems may require developing data preprocessing software to minimize bad data processed by the IS and providing self-correction within the IS. Often, the interface to the IS is developed in conjunction with a general user interface upgrade, since expert system tools can be the first opportunity to use a graphical user interface in the space operations support environment. The intelligent system displays must be integrated with other displays. Failure to design the IS for coordination with the human operator is another design problem affecting its integration into operations. Designing for coordination requires avoiding unnecessary interruptions or interference in operator activities. Task handover and task dependencies (including required information exchange) should also be considered during design.

We have summarized the common intelligent system design problems observed during the case study. These problems can significantly impact IS reliability and usability. In table 2-1, we categorize IS design problems by their impact to reliability and usability, and provide recommendations for solving these problems. Reliability is a critical design issue, since an unreliable and uncontrollable intelligent system can impact the safety of space systems. Intelligent systems that do not perform reliably cannot successfully provide real-time decision support. Thus, solving those problems affecting reliability should be of first priority to the IS designer. The second impact is related to IS usability. Intelligent systems that are difficult to use have an increased risk of user rejection and user errors. Solving problems affecting usability should improve the chances of intelligent system success.

---

[2] The risks of correction include causing irrevocable changes, making errors during correction, and introducing unexpected effects. The risks of reallocation arise from users not sufficiently aware of situation to respond correctly, not adequately trained, and overworked or distracted by assuming IS responsibilities.

Table 2-1. Summary of Intelligent System Design Problems and Recommended Solutions to Those Problems

DESIGN PROBLEMS	RECOMMENDATIONS
Problems Affecting Reliability	
1. IS not designed to support human operator in supervising IS activities and recovering from IS errors	• Assess risks of correcting or compensating for IS error • Provide user correction or compensation for IS error • Support managing monitored process independent of IS • Support dynamic task allocation (human backs up the IS)
2. Unrealistic assumptions about space operations • bad or unavailable data • unexpected situations	• Preprocess data • Design IS for self correction • Design IS for supervised recovery (see 1 above)
3. IS activities, reasoning strategies, and capabilities misunderstood and often overestimated by human operator (i.e., a magical IS)	• Provide dynamic information and feedback about ongoing IS activities and strategies • Provide access to static information about IS capabilities • Emphasize information relevant to managing the IS • Show situation as it develops, including IS activities
Problems Affecting Usability	
1. Common practice in user interface design increases user workload and fails to provide the most usable information • message lists and schematics • explanation	• Identify information requirements to support domain tasks • To avoid overload, present what is relevant to ongoing task • Present situation as it develops, including IS activities • Replace explanation with shared understanding of situation
2. IS not integrated with support system	• Integrate with operational data sources • Integrate with other support displays
3. IS not designed for coordination	• Avoid interrupting user or interfering with user tasks • Coordinate changes in task allocation • Identify dependencies between tasks • Provide feedback about ongoing IS activities

Using this design guidance should improve safety and reduce cost. Building reliable intelligent systems reduces safety threats due to IS errors. Building usable intelligent systems reduces the potential for operator error and improves user acceptance. This reduces the chance of the IS not being used, and minimizes costly redesign of the IS. The results are safer operations using the IS and reduced cost of building the IS.

2.4 The Role of Information in Designing Team Players

Now that we have described what it means for an intelligent system to be a team player and characterized the information problems that can prevent it from being a team player, we will describe how to design systems that are team players. Let us start with an example of a bad design, an intelligent system that is not a team player.

> *Example*: Suppose that a user's task is to detect event Y, which occurs when sensor A is bad and switch B is off. The intelligent system displays the current status of sensor A and state of switch B. If noisy data causes a change in the displayed value from either sensor A or switch B before the user looks at the display, the user can miss event Y.

On the surface, the problem with this system appears to be caused by "bad" user interface design (e.g., overwrite the display of data from sensor A and switch B before event Y can be detected). A closer look, however, reveals that the so-called bad user interface is merely a symptom of an underlying information problem (e.g, the information of interest is Y, but the IS does not detect and display Y). Specifically, the intelligent system is not providing the right information at the right time. These information problems can affect the ability to perform both domain tasks related to the monitored process and user supervisory tasks for the intelligent system. An understanding of both of these types of tasks is necessary to determine what the"right" information is and when it should be provided. We have defined *information requirements* as describing the information that must be exchanged between the human and computer when they interact to perform both of these types of tasks.

Information requirements are identified by considering how the user will interact with the support system, including the intelligent system, to perform tasks (i.e., based on a specification of expected operations using the intelligent system). Thus, information requirements include operational considerations early in system design, a common recommendation in the design of space support systems that is rarely applied. Representative scenarios are defined that specify the expected operations using the intelligent system. These scenarios are then evaluated to identify the information that must be exchanged between the human and the intelligent system. Techniques for evaluating operational scenarios include exercising prototypes or storyboards.

Why does defining information requirements result in an intelligent system that is a team player? Because these requirements are based on a specification of the user's tasks during operations with the IS (both domain and user supervisory tasks). A task-based approach to requirements definition ensures that the monitored process can be managed, and that the resulting IS can be managed (i.e., it is reliable, and directable by the human coach) and that its activities are coordinated with those of the operator. Information requirements provide an informed means of selecting what should be represented and presented, as well as when it should be represented and presented. This guarantees that all the needed information, and only the needed information, is provided, which solves the issue of effective communication (and the associated concerns of a magical IS and information overload). Additionally, information requirements provide a more rigorous and objective means of evaluating the design than the usual evaluation approach in which the design is

good if the users like the way it looks (sometimes called design by "Mikey likes it"; Abbott, 1992).

Information requirements affect the design of both the intelligent system (i.e., what information to *represent*) and its user interface (i.e., what information to *present*). Thus, user interface design and intelligent system design are not independent efforts, but aspects of a single development process. Considering human-computer interaction as part of system development integrates user interface design into overall system design. It also necessarily involves users early in system design to describe the task and assist in identifying information requirements.

We make a strong distinction between HCI design and user interface design. Human-computer interaction design focuses on the information exchanged between the user and the computer when performing the task that the software system is intended to support. In other words, the representation of information (i.e., the information message or meaning) is the product of HCI design. The user interface design, on the other hand, focuses on the display and control software and hardware (i.e., the information media) that mediate information presentation and dialog with the user. One of the dangers of confusing HCI design with user interface design is the tendency to overlook task-level information needs (e.g., information needed to make a good decision), while focusing on physical interface characteristics and low-level interface functions. Based on this distinction, HCI expertise is needed early in system development to define the information required to perform tasks. The more traditional user interface expertise is needed to design for information presentation and dialog with the user, which may include early development of user interface design concepts, in the form of prototypes or storyboards.

In the next section (section 3), the activities performed to define information requirements are discussed. Section 3.4 addresses information requirements for managing the monitored process, and section 3.5 addresses information requirements for managing the intelligent system.

Section 3
Guide for Designing Human-Computer Interaction

During the case study described in the Tech Memo (Malin *et al.*, 1991), we frequently asked intelligent system designers if they had used any design guidelines when designing the intelligent system. Their answers ranged from "I tried but I couldn't find what I needed" to "I just didn't have time". The problems that they experienced are not uncommon, and arise from difficulties in locating design guidance relevant to the designer's needs. What the designer needs can be expressed in a variety of ways, such as good design examples, solutions to typical problems, or assistance in performing design tasks. Thus, multiple indices into guidance are needed. But a paper document inherently has a single, fixed organization.

This section provides an alternate organization for the material in the Tech Memo, using the major intelligent system analysis and design activities defined in section 6.1 of the Tech Memo (see figure 3-1). Thus, the designer can use current activities (e.g., defining requirements) as an index into the HCI design guidance. We are not attempting to describe the analysis and design process in any detail, or to recommend a fixed ordering of these activities, but to indicate important HCI considerations relevant to each activity. What has resulted is a guide for considering human-computer interaction in the design of intelligent systems.

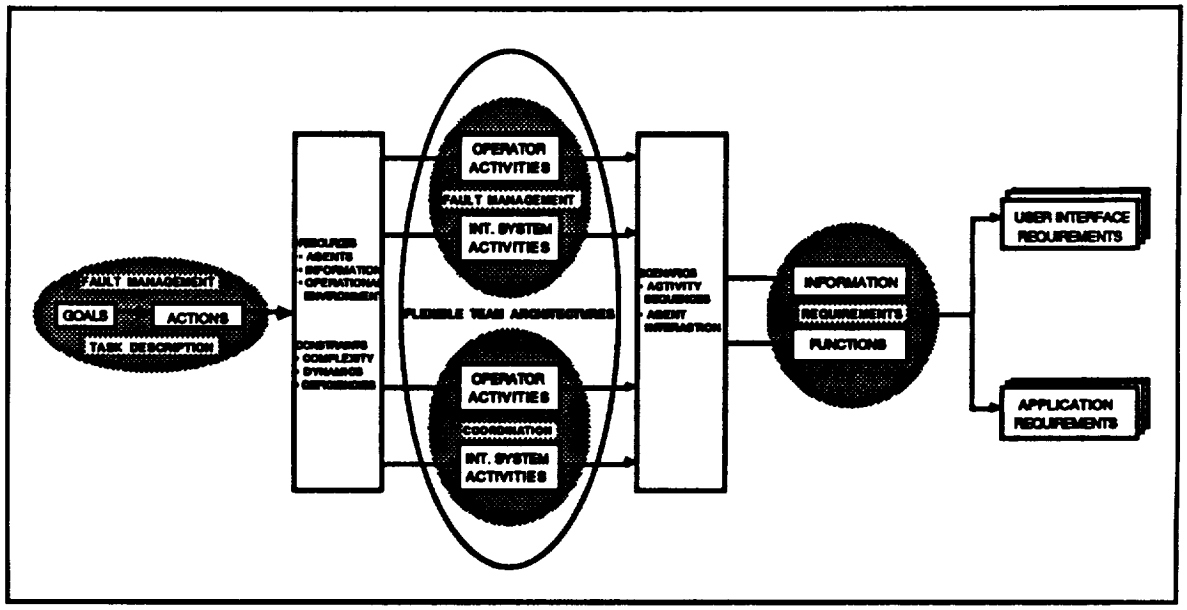

Figure 3-1. Mapping from Task Description to Information Requirements (Malin *et al.*, 1991)

This section is written for developers of an intelligent system and its user interface. This may include personnel with widely varying expertise, including knowledge engineers, intelligent system programmers, human factors experts, domain experts, and intelligent system users. The topics covered in this section are listed below:

- Preparation for Design
- Analysis of Agent Tasks and Team Architecture
- Evaluation Using Scenarios

- Information Requirements for Process Management
- Information Requirements for Intelligent System Management

Notice that this section only addresses the early phases of analysis and design, with the goal of developing information requirements for use during design. Thus, specific user interface design guidelines are not discussed.

To assist the reader in relating the topics covered in this report to the appropriate sections in the Tech Memo, we open the description of each topic with a table listing the sections of the Tech Memo that are covered. We have also included a list of the problems addressed in these sections. The reader should refer to the Tech Memo for design examples illustrating these problems and their solutions.

3.1 Preparation for Design

We have identified a number of considerations when preparing for design. These considerations are concerned with the formation of a design team and the selection of design techniques. Design preparation should be made with an awareness of how the intelligent system design will be integrated with the overall support environment. In this section we discuss the design team. We describe some common design techniques and when they are useful. We describe an intelligent system design perspective that is integrated with the target support environment and that accommodates both system upgrade and evolution.

3.1.1 Design Team

Reference to the Tech Memo
Related Section
6.2 Design Team
Problems
Communication within Design Team (6.2.1)
Selecting Members of the Design Team
Ensuring Communication between Design Team Members
User Participation in Design (6.2.2)
* Involving the Users in Design

Just as there is a need to coordinate the space operations support team for effective interaction, there is also a need to coordinate the design team (typically all human) to achieve integrated, consistent design products. Communication between members of the design team is central to coordinated design. Information requirements represent a single repository of design information useful to both the intelligent system designer and the user interface designer. As such, they can provide an effective mechanism for communication within the design team.

The constituency of the design team should be multi-disciplinary. Expertise about the space system and its operations (e.g., user or design engineer), software implementation and integration (e.g, intelligent system and user interface technologists, knowledge engineers, computer hardware and software engineers), human-computer interaction, and human factors/user interfaces are all required. This expertise can be provided in a variety

* Asterisk indicates especially important problems.

of ways, depending upon the nature of the development project. For example, when upgrading an existing system (such as adding an intelligent system to Space Shuttle operations), an experienced user may also be the space system domain expert. For some of the Real Time Data System (RTDS) prototypes[1], the users also implemented the software, with the support of computer hardware and software engineers. Conversely, when developing a new system (such as intelligent software for Space Station) domain knowledge is likely to reside with design engineers, who may eventually become the software implementers as well. Operations experience can also be more difficult to identify for a new system. It may be necessary to rely on related operational experience (such as provided by a Space Shuttle flight controller for a Space Station project) or an experienced design engineer who is familiar with space operations.

It is important to distinguish between HCI expertise and user interface expertise. HCI expertise is concerned with the information *message* (or content), while user interface expertise is concerned with the information *media* (or presentation). HCI expertise is especially important during the analysis and early design phases, to assist in defining information requirements on which the design will be based. Both types of expertise are necessary to design for effective information exchange and human-computer interaction.

Design participation by a user, or other representative of operational expertise, is necessary to ensure that the designed system will support user tasks. User participation can be both direct and indirect. Direct user analysis and design activities include describing the tasks, defining operational scenarios, and, in some cases, development of prototypes for requirements definition. Indirect user design activities are primarily forms of design product evaluation and review, including demonstration and hands-on testing (both side-by-side and off-line, stand-alone). The user's role in the design team should be clearly defined and should focus valuable, and often limited, user resources on aspects of analysis and design that are best addressed by them (i.e., user task description, operational scenarios).

3.1.2 Design Techniques

Reference to the Tech Memo
Related Section
6.1.2 Techniques
Problems
Prototypes (6.1.2)
Testing the HCI Design
* Testing the Design under Realistic Conditions
Storyboards (6.1.2)
* Providing Techniques to Identify Information Requirements
Documenting the Design Constraints

Prototyping is by far the most common technique used for intelligent system analysis, design, and development, including HCI design. Iterative prototyping is used for a variety of design tasks, ranging from full-scale demonstrations of system capability integrated into an environment to small, stand-alone prototypes for testing specific design concepts. A second technique, storyboarding, is less commonly used but can be effective for

[1] The Real Time Data System (RTDS) prototypes were developed for use by flight controllers in the Space Shuttle Mission Control Center.

identifying and testing HCI design concepts. The two techniques can be combined by generating storyboards using electronic prototyping tools. The products of both techniques can include requirements, usable design fragments and concepts, or even occasionally reusable code.

These techniques have a number of characteristics in common. Both techniques can be used to generate information requirements, although derivation of requirements from prototypes and storyboards is currently a manual task. The products of both techniques are amenable to evaluation and refinement using operational scenarios (see section 3.3 of this report for additional information). Note that use of prototyping tools may introduce design constraints peculiar to the tool. Since such tool-related constraints should not become part of the system design, it is important that constraints be documented, including the source of constraints.

3.1.3 Design for Upgrades of Existing Support Systems

Reference to the Tech Memo
Related Section
6.3 Integration into the Support Environment
Problems
* Integrating Intelligent System Design into Design of Support System
  Avoiding Negative Impacts Caused by System Upgrade
* Integrating a New or Different Technology into an Existing Environment
  Considering System Evolution during Design
  Matching the System Design to the User's Task

Intelligent systems developed for process monitoring and control, such as space operations or nuclear operations, are often elements of a much larger support system, consisting of both intelligent and conventional, non-automated software. A recommended philosophy of design for such complex, multi-part systems is to embed individual element design into the design of the entire support system. When the individual element is an upgrade to an existing support/multi-part system, its design should be influenced by operations within the existing support framework. Since applying new technology often changes the nature of human tasks, it may be necessary to significantly alter, or redesign, operations. Some typical changes in operations include the following:

- human supervises the intelligent system performing a task instead of directly performing the task
- automation of written logs and manual computations
- electronic manipulation of graphical forms

Upgrading elements of a multi-part system poses some unique challenges. Human tasks can be complicated by inadvertently removing useful tools and methods, or by increasing the human workload. Changing the human task can result in lost opportunities to develop or maintain expertise while performing the job. Gradual integration of new technical capability with the old appears to be an effective way of accomplishing element upgrades and meeting some of these challenges. Multiple, small, incremental changes were frequently used to upgrade systems in the case study. Also, gradual replacement of the old provides time to evaluate the new capability thoroughly, which reduces technical risk and improves user acceptance. Such an approach can include periods of operating both the old and the new capability side-by-side.

Existing manual operations and off-line support tools are an important source of HCI design ideas when upgrading an existing system. Such derived designs are familiar to users and often represent considerable design refinement when developed and used by multiple users over a period of time. Such designs can also be, however, useless artifacts of the previous design (an outmoded technology). To assist in making this trade-off between the useful and the artifactual, the design should be evaluated for consistency with the planned use of the information represented by the design.

It is important to design support systems to ease future upgrades. Designing for system evolution includes such considerations as accommodating transitions to alternate information/data sources or automation of manual functions.

3.2 Analysis of Agent Tasks and Team Architecture

An early phase in developing intelligent systems should be analyzing what tasks need to be done and how the human-intelligent system team will do these tasks. Such an analysis of the tasks and the structure of the team is important in designing for effective human-computer interaction. Few of the systems we have studied, however, performed such an analysis[2]. One of the difficulties is that an intelligent system development methodology for such an analysis doesn't exist. We are currently investigating such a methodology and the related analysis methods.

We have partitioned the analysis of agent tasks and team architecture into three parts. First, the tasks to be performed are described (*task description*). Next, these tasks are allocated to either the human, the intelligent system, or both (*task allocation*). Finally, the way that the human and the intelligent system interact based on these allocations is determined and evaluated (*team architecture*). In this section we describe our work to date on each of these parts of the analysis. Considerable work remains to be done and, when possible, we have pointed out areas of needed research.

3.2.1 Task Description

Reference to the Tech Memo
Related Sections
- 3.1 Fault Management Task Description
- 3.3.1 Types of Agent Activities
- 3.5 Fault Management Activities and Information
- 6.1.1 Task Description

A task description is defined as a specification of the team's (human and intelligent system) tasks in terms of goals and the actions required to achieve those goals. Each task consists of one goal and all the actions required to achieve the goal. Thus, a task description can be represented as a goal hierarchy, in which top-level goals are typically subdivided into sub-goals. We distinguish a task description from familiar forms of task analysis, such as the GOMS task analysis (Card, Moran and Newell, 1983). The task description is derived much earlier in system development (during system analysis prior to design) and is, thus, at a much higher level than the keyboard-level description of a GOMS analysis.

[2] The one exception is the Extended Real-time FEAT (ERF), an advanced automation system for use in the Space Station Control Center (SSCC) (Clark *et al.*, 1992). The ERF developers did an early evaluation of tasks, although they did not do a complete analysis as described in this section. ERF is one of the cases we are investigating at this time.

A task description looks different because it is used for different things. Specifically, a task description is used to describe the tasks to be allocated and to specify the goals to be achieved by the space operations support team - not to design the user interface. It is also used in designing the team architecture, since the activities in this architecture are derived from the goals and actions of the task description. Additionally, shared tasks (i.e., a task where multiple agents work to accomplish a single goal) and task dependencies (represented in the levels of the goal hierarchy) are used to locate points where agents must coordinate their activities.

In the Tech Memo we characterized the top-level fault management goals and tasks (summarized in section 1.3), and described the kinds of activities and information occurring in fault management[3] (see appendix D of the Tech Memo). A task description for a specific intelligent system would address some subset of these activities and information (e.g., most systems in the case study only performed monitoring and assessment activities). Defining typical, expected operational scenarios using the intelligent system can be useful in both describing the task and evaluating its completeness (see also section 3.3).

3.2.2 Task Allocation

Reference to the Tech Memo

Related Sections

- 3.2.1 Characteristics of Aerospace Operations
- 3.2.2 Characteristics of Agents and Teams
- 4.1.1 Multi-tasking and Dynamic Task Assignment

Problems

Task Allocation (4.1.1)

- Task Allocation with Task Sharing
- * Maintaining Operator Control of Monitored Process (Requires Dynamic Task Allocation)

Dynamic Task Assignment (4.1.1)

- * Specifying the Architecture of Agents and Activities Handling Alteration of Planned Activity Sequences

After specifying a task description, it is necessary to allocate these tasks to human operators, intelligent systems, or both. *Task sharing* describes situations where a task is allocated to both the human and the intelligent system. Task sharing is accomplished by providing a nominal allocation with an alternate, backup allocation (i.e., a form of redundancy, in which one agent backs up the other agent). Changes from the nominal to the backup allocation requires that the IS be designed for dynamic task allocation. Such dynamic allocation is necessary to ensure that critical tasks can be performed by the human when IS errors occur. In general, human operators are considered more reliable than intelligent systems in the space operations environment (see section 2.2). Thus, the human operator should share IS responsibility for critical tasks, and the human should be the final decision maker in such tasks (consistent with the human role of coach described in section 2.2). It may be necessary to dynamically alter task allocations during operations to

[3] There are three types of fault management activities: (1) monitoring and assessment, (2) planning and dynamic replanning, and (3) intervention and control. A fourth type, coordination activities, is necessary for effective team interaction. Types of information supporting fault management activities include (1) dynamic data, (2) mission specific information, (3) baseline operations information, and (4) design knowledge.

accommodate workload imbalance, performance problems, or unanticipated situations. Alternative task allocations should be provided to accommodate these situations, including a means for the human to reassign tasks. See section 3.2.3 for a related discussion on specifying dynamic task allocation as a team architecture.

The tasks specified in the task description should be allocated based on the available resources and the design constraints. Resources include support hardware and software, agents (humans and intelligent system), and information. The support software and hardware in which an IS is to be embedded can also be a design constraint on task allocations to the IS (e.g., limited CPU or memory can constrain what the IS is able to do in real time). Policies can constrain allocations as well (e.g., intelligent systems will not issue commands). The way intelligent systems are used for space operations constrains how tasks are allocated to the IS (see also section 2.1).

In addition to the constraints that the task allocation must satisfy, there are a number of suggestions for improving allocations. The developer should allocate tasks based on an understanding of the strengths and weaknesses of both humans and intelligent systems. A good rule of thumb is to allocate IS tasks to amplify human strengths or compensate for human weakness. Humans are good at recognizing constant patterns in varying situations, unless the patterns form at rates outside the limits of human perception (either too fast or too slow). Thus, an IS can be used to compensate for human perceptual limitations by detecting patterns that vary outside these limitations (e.g., slowly developing trends). Humans tend to reason using cases or analogies from their experience (Andriole and Adelman, 1991). They perform reasonably well at the boundaries of their knowledge, unlike intelligent systems which tend to make errors outside the boundaries of their knowledge. Intelligent systems are also subject to performance limitations caused by CPU loading or memory limitations.

When making decisions and solving problems, humans have a propensity to focus on a subset of the available information. They selectively attend to information consistent with their current hypothesis, ignoring information not considered relevant to the situation (*confirmation bias*; Andriole and Adelman, 1991). They tend to focus on one hypothesis quickly and do not give equal consideration to all alternatives (Arkes and Hammond, 1988). They can even forget evidence to the contrary of a favored hypothesis (*favorable memory error*; Arkes and Hammond, 1988). Tasks should be allocated to compensate for human propensities (e.g., the IS can remind the operator of evidence against a favored hypothesis or evidence supporting other hypotheses).

Finally, tasks should be allocated so that human operators receive adequate on-the-job training to ensure their continued ability to perform fault management tasks as well as supervise the IS. This expertise includes understanding task goals and the corresponding procedures for accomplishing them, developing mental models of relevant space systems and processes, and acquiring skills for interpreting data and responding to fault situations (Bloom, 1991).

We have described the constraints on task allocation and made a number of suggestions for improving these allocations. It will not usually be possible to satisfy all of these considerations. Performing task allocation is essentially making a trade-off between competing considerations. When performing this trade-off, we recommend selecting allocations that minimize risk and maximize IS reliability. Significant task sharing is required to maintain human operator control of the monitored process and the intelligent system, and to guarantee adequate on-the-job training. Additional research is needed to determine appropriate task allocations for intelligent systems and humans working in teams.

3.2.3 Team Architecture

Reference to the Tech Memo
Related Sections
3.3.2 Interaction Between Agents
4.1.1 Multi-tasking and Dynamic Task Assignment
Problems
Dynamic Task Assignment (4.1.1)
* Specifying the Architecture of Agents and Activities
Handling Alteration of Planned Activity Sequences
Distinguishing Active Mode of Operation
Handover (4.1.1)
Orienting In-coming Operator at Handover
Interruption of Operator (4.1.1)
* Handling Interruptions and Suspended Activity

An important aspect of building reliable systems is providing alternative ways of achieving goals when planned activities are not effective (e.g., human backing up the IS when it malfunctions; see section 2.2). The need to find alternative ways of accomplishing goals can arise for a variety of reasons, including (1) degraded agent performance due to workload or error, (2) failure of the space operations support team to coordinate their activities, (3) goal or plan changes due to failures in the domain system being monitored (the monitored process) or environmental perturbations or accidents, or (4) unexpected events or contingencies. Designing the IS to accommodate alternatives means providing the ability to dynamically alter the normal sequence of activities, modify activities (including add and delete them), and reassign activities. The approach most compatible with current IS capability is dynamically reassigning activities.

Handover is the space operations term for changing the allocation of a task from one member of the space operations support team to another. Typically, handovers between human operators occur at shift changes. They may also occur during busy periods to redistribute workload among team members. Handover also plays a part in coordinating human-intelligent system teams. For supervised tasks, the human must be prepared to take over from the IS if a problem occurs. For sequential tasks, task responsibility alternates between the human and the IS to accommodate task dependencies or to prevent task interference. To respond effectively to handover, the team member assuming responsibility must understand the current situation, including the events leading up to that situation, and what responsibility is being handed over (i.e., what goals are to be achieved).

Coordinating human and intelligent system activities during multi-tasking, dynamic task assignment, and task sharing requires consideration of task dependencies (timing, events, behavior, and resource) and interference points. Task dependencies are accounted for by ordering the activities[4] performed to accomplish the task and by handing over task responsibility between agents. It is not tractable to completely specify all possible activity sequences in a complex domain. Instead, the developer must specify a range of allowable task allocations for each agent and identify activities within these allocations where agents

[4] An activity is an action that has been assigned to a specific agent. A task will consist of a sequence of activities performed by one or more agents. Types of activity ordering can include sequential, parallel and independent, simultaneous and dependent, preferred but not required, and non-interfering.

must interact (e.g., information is required from the IS before the task can continue) or where agents must not interfere (e.g., don't interrupt a critical operation with trivia). We call this specification a team architecture. The team architecture should include both human and IS agents, their activities, and the information that they exchange. This architecture should support nominal and off-nominal domain tasks for managing the monitored process, and user supervisory tasks for managing the intelligent system. It should also include provision for contingency situations[5].

A team architecture can be used by the developer for many purposes. First, the team architecture defines alternative ways to accomplish tasks. These alternatives identify where (and to whom) tasks can be reassigned, and are especially important in specifying modes of operation and in locating handovers of responsibility between agents. Second, the team architecture defines points of coordination between the human and the intelligent system. Coordination points include both activities where dependencies must be accommodated and activities where interference must be avoided (e.g., unnecessary interruptions). These points of coordination represent regions of tightly constrained activity which require close attention from the developer to ensure that effective operation is possible. The explicit identification of coordination points is also necessary to specify information requirements for coordination activities. These requirements can not be identified by looking at the domain task. Finally, the team architecture can be useful in evaluating task allocations for both task loading and feasibility of candidate allocations. Note that to perform such an evaluation, the developer must consider information about all activities performed by both the human and the intelligent system, including those independent of the user interface (e.g., human manual tasks). Additional work is needed in specifying the structure of a team architecture and methods for determining the team architecture.

A common way of implementing dynamic task assignment is providing alternative task allocations in the form of multiple modes of operation[6]. For example, an IS may have different control modes that vary its level of autonomy in executing activities. Modes enforce task reassignment by constraining what the IS can do in a particular mode. Modes can also provide a context for the presentation of information. Intelligent system capability, information, and style of interaction can all vary by mode.

When the intelligent system has multiple modes of operation, the operator should be trained to know the range of possible modes (task assignments), when they are used, and how to change them. When using a multi-mode IS, the operator must know what mode is currently active and what the user can (and cannot) do in that mode. If the IS modes are not well understood by the operator and clearly distinguished during operations, the IS can be perceived as having mysterious behavior (e.g., the operator is disoriented by "missing" information when information available in one mode is not available in another mode), or its actions can be completely misinterpreted (e.g., a hypothetical failure being evaluated in a what-if scenario is misperceived as an actual failure; see section 2.1 for a discussion of magical intelligent systems).

Interruptions are a common way that intelligent systems interfere with human activities. Such interruptions can be routine (e.g., notifying the operator that a background task is complete) or can indicate that a problem has occurred (e.g., alerting the operator about a failure in the monitored process). When an activity is interrupted, it must be suspended while the human decides how to respond (which activity to pursue, the new or the

[5] Contingencies are situations in which the unexpected (and thus not planned or designed for) occurs and which introduce the potential for harmful impacts to safety or mission objectives.

[6] Modes of operation are sets of possible activities for each agent, not all of which are performed under normal circumstances (Malin *et al.*, 1991).

ongoing). If the human pursues the new activity, the ongoing activity may be suspended temporarily (e.g., while the human does a more important task), handed over to another team member (i.e., task reassignment), or aborted. If the suspended task will be resumed later, it is desirable to retain the work in progress.

Intelligent systems used during space operations should avoid unnecessarily interrupting the human operator. When interruptions are necessary, the IS should assist the human in responding to the interruption. There are a number of strategies for assisting the human in responding to interruptions. If interruptions are expected to be infrequent, or ongoing activities are not critical, the IS can take no additional action, merely providing the human with all the information all the time (i.e., interrupt and let the human decide). Alternately the IS can limit interruptions. Message priority is a common way of limiting interruptions of the operator by new information. Priorities are assigned based on the importance of the message to safety and mission objectives. The operator can delay handling all but the high priority messages (typically safety related). Another way to limit interruptions is to queue all new requests or messages, and merely alert the operator that new requests have been made or information provided. There is a need for better ways to manage the presentation of new information and activity requests that minimize interruptions to the operator.

3.3 Evaluation Using Scenarios

Reference to the Tech Memo
Related Section
- 3.4 Fault Management Scenarios
- 6.1.2 Techniques
- 6.2.2 User Participation in Design

Problems
Prototypes (6.1.2)
- Testing the HCI Design
- * Testing the Design under Realistic Conditions

Storyboards (6.1.2)
- Providing Techniques to Identify Information Requirements

User Participation in Design (6.2.2)
- Involving the Users in Design

Evaluation of analysis and design products using operational scenarios representative of typical or expected operations is an important and recurring activity in the intelligent system design and analysis phase. Many products of these activities can be evaluated, including task descriptions, task allocations and team architectures, requirements, and design concepts. These products may be in the form of documents, diagrams, storyboards, or prototypes. Such an evaluation is done to determine how well the concepts embodied in the product supports the user performing tasks (e.g., system correctness, usability). In this section we discuss the definition of typical fault management operational scenarios, including factors that can complicate operations. We also discuss the evaluation of analysis and design products using operational scenarios.

In section 3.4 of the Tech Memo, we describe a generic fault scenario. This scenario is simplified to clarify the activity sequence. This description consists of the steps required to manage the monitored process from the time that a failure is detected until the failure is repaired. These steps are listed below:

- detection of failure
- safing
- mission impact assessment and accommodation
- diagnosis
- testing
- recovery

Within each step, the activities required to complete the step are listed. To see how these steps relate to fault management tasks, see sections 2 and 3.2 of this report.

Different types of scenarios are needed at each point in the analysis and design process, because the purpose of the evaluation is different at each point in the process. Scenarios can represent a wide variety of operational "views" (i.e., emphasize different aspects of operations, such as whether the scenario assumes that sensors always provide perfect data). Typically, higher fidelity scenarios (i.e., scenarios that more accurately represent operations) are needed later in the process.

Higher fidelity scenarios include complicating factors in addition to the fault being managed. Anomalous behavior of the monitored process is not always caused by a failure in that system. Apparent failures may actually be caused by bad input data (e.g., sensor failure or data transmission error) or by a misconfigured system. Anomalous behavior of the space operations support team can also cause anomalies in the monitored process. Team anomalies include human errors or omissions, intelligent system failures, or failure to coordinate the activities of multiple team members (human or intelligent system). Loss of data, due to unavailable or failed data transmission/acquisition systems, can result in data ambiguity and the inability to monitor the impacts of fault management activities onboard the space vehicle (e.g., system response to fault management activities or crew execution of fault management procedures).

There are additional factors occurring in the space operations environment that can complicate the ideal fault management scenario. Operations unaffected by the failure continue in parallel with fault management activities. Activities related to these nominal operations can interrupt fault management activities. The cause of the anomaly may be intermittent, resulting in erratic anomalous behavior. The anomaly may also be due to combinations of causes (e.g., multiple failures), which can difficult to recognize and disentangle. For complex systems, it is difficult to anticipate all system failures, and unanticipated failure situations can occur for which there are no predefined fault management procedures. Fault management activities will also vary based on the criticality of the lost functional capability or hardware item, and the availability of redundant capability.

The designer should use scenarios that adequately represent the operational aspects of interest in the evaluation. Some considerations when selecting (or designing) a scenario for use at different points in the analysis and design process are listed below:

- Task Description
  Simple scenarios are needed that outline typical operations (i.e., nominal operations and some representative off-nominal situations). These scenarios are used to evaluate the task description for accuracy, consistency, and thoroughness.

- Task Allocation and Team Architecture
  Scenarios should emphasize situations where the human and intelligent system interact closely, including examples of task sharing, tasks with dependencies,

and dynamic task allocation. These scenarios are used to evaluate the task allocation for coordinated interaction and balanced workload.

- Early Prototypes
  Scenarios should reflect more complexity in the operational environment and include such complicating factors as imperfect sensors and data problems that distort the operator's view of the situation (as discussed previously). Early prototypes are evaluated for information deficiencies (i.e., missing or misrepresented information) and interaction difficulties that affect both the intelligent system and user interface specification.

- Mature Prototypes
  High fidelity scenarios (based on accurate simulations or real data) are needed that include very specific anomaly situations (i.e., case testing), but do not represent the breadth of operations (i.e., focus on anomalies result in partial scenarios where much is implicit). Mature prototypes are evaluated for accuracy, completeness, and robustness of the intelligent system, as well as ease of use and appropriateness of representation for the user interface. Integration with other specific elements of the support system can also be tested at this point.

Common practice in evaluating prototypes is to perform two types of testing: (1) off-line testing, and (2) in-line, side-by-side testing with an existing support system (e.g., RTDS parallel operations). Both types of testing emphasize evaluating specific anomaly situations, but they vary by the degree of fidelity and integration. Often focused, stand-alone tests are conducted early in design on products defining small portions of capability (e.g, proof-of-concept prototype, user interface design concepts). Integrated, in-line testing of larger portions of capability tends to occur later in design. The data used for testing vary with respect to the purpose of the test (e.g, rough simulation for early definition, high fidelity simulated or real data to test functional accuracy or integration). See section 3.1 for a discussion of user roles in evaluation using scenarios (e.g., hands-on tests, reviewing design products).

3.4 Information Requirements for Process Management

The *raison d'etre* of the space operations support team is to ensure that the monitored process[7] is performing as expected. This includes monitoring and controlling the execution of procedures as well as responding to behavior anomalies. In this section we discuss the information required to manage the monitored process. In section 3.4.1, we address monitoring and controlling the monitored process, especially the information needed to manage alarms. In section 3.4.2, we describe how the space operations support team responds to anomalies and what information they need to do this task.

[7] The monitored process is the domain system being monitored and controlled; see section 1.3 for further information about this term.

3.4.1 Monitoring and Managing Process Alarms

Reference to the Tech Memo
Related Sections
- 4.2.1 Alarm Management
- 4.3.1 Interpretation of Information

Problems
Interpretation of Alarms (4.2.1)
- Indicating Alarm Accuracy
- Distinguishing Severity of Alarms
- * Relieving Operator Overload due to Multiple Alarms

False Alarms (4.2.1)
- Managing False Alarms due to Bad Data
- Managing False Alarms due to Transient or Intermittent Behavior
- * Differentiating Misconfiguration from Failure

Redundant Alarms (4.2.1)
- Relieving Operator Overload due to Redundant Alarms

A significant portion of the operator's time is spent monitoring the behavior of the monitored process. In fact, most intelligent systems in the case study were built to alleviate some of the human's monitoring and vigilance responsibilities. To monitor for anomalies in the monitored process, the operator must have some expectation of what its behavior should be (based on both plans and operator experience). Typically, an *alarm* is annunciated when behavior is not within the envelope of nominal behavior[8]. Multiple alarms may be issued shortly after the initial alarm due to failure propagation into other systems and redundant alarms. Additionally, anomalies can have serious implications for safety and mission objectives, and can impose hard timing constraints. Thus, managing information from multiple alarms increases operator workload just at the time when the operator can least afford it.

The space operations support system, including the intelligent system, should be designed to provide the right information to the operator for managing alarm information and understanding the monitored process situation. Presenting alarm information to improve understanding of situation requires focusing operator attention on what is diagnostically important, and quickly and clearly indicating the diagnostic content and relationships among alarms. It may be necessary to illustrate events and activities leading up to the anomaly as context for interpreting an anomaly.

The complexity and uncertainty of space operations makes handling false alarms (i.e., inaccurate indications of anomaly) an inevitable and important part of managing the monitored process. Handling false alarms is also important in designing a reliable intelligent system, since false alarms can cause IS errors. Managing false alarms shifts the operator's attention from managing the monitored process to managing the view of the monitored process (i.e., how accurately does the operator's view into the monitored process via data represent what is actually happening). Noisy, unreliable data and transient information at state transitions can cause false alarms. Failed sensors can be difficult to detect, leading to false alarms. Complex operations requiring configuration changes can result in misconfigurations erroneously detected as alarm conditions. Operating

[8] The envelope of nominal behavior is usually expressed as parameter limits, or sets of limits specific to an operating mode or configuration

characteristics can differ from the designed characteristics (e.g., differ among similar devices, change with use and passage of time for one device), resulting in false alarms.

It is important to minimize false alarms and, when they do occur, to provide the operator with the means to handle them. Noisy data can be filtered prior to issuing alarms, or alternate data sources can be provided. Alternate data sources are also effective in detecting failed sensors. Transients may be separated from slowly developing anomalies by waiting for them to settle before issuing an alarm. To assist in separating transients from quickly developing anomalies, the operator can be alerted when in a time period where transients are likely to occur (e.g., state changes). To avoid false alarms due to configuration, configuration can be confirmed prior to issuing alarm. It is also possible to modify operating characteristics in real-time (e.g., loosen limits) to accommodate drift in these characteristics over time. A note of caution - if any of these alterations to nominal configuration are performed, the operator must maintain awareness of the changes and incoming operators must be informed at shift handover.

The risks associated with space operations increase the potential impact of mishandling alarms. Redundant alarms are often provided to reduce the risk of a false alarm and to provide alternate alarms when a failed sensor effectively disarms an alarm. If redundant alarms agree, the operator's confidence that an anomaly has occurred is increased. Redundancy can complicate operations, however. More alarms mean more information for the operator to process. And if alarms disagree, additional effort is required to resolve and interpret inconsistencies. Redundant alarms can also cause an intelligent system to issue duplicative messages.

Techniques for assisting operators in managing alarms include suppressing redundant, false, or irrelevant alarms, assessing confidence in alarm accuracy (e.g, reliability of source of alarm; consistency among related alarms), and synthesizing or combining alarms (both algorithmic and visual composites). The source, quality, and validity of the data triggering the alarm are also useful in assessing trust in alarm accuracy.

The intelligent system can both be a source of false alarms and can be led astray by false alarms. An incomplete or incorrect knowledge base can cause the IS to erroneously detect an anomaly. Processing false alarms in data can cause IS error. The effect of false alarms on IS behavior can be minimized by designing for reliability (i.e., providing the operator with the capability to either redirect the IS or to disable the IS; see also section 2.2).

3.4.2 Responding to Anomalous Situations in the Monitored Process

Reference to the Tech Memo
Related Sections
4.2.2 Critical Diagnostic Information
4.2.3 Unanticipated Situations and Workaround
Problems
Visibility into Monitored Process (4.2.2)
Maintaining Awareness of Monitored Process Situation
Evaluation of Consequences (4.2.2)
* Support for Evaluating Consequences of Events & Team Activities
Mission Impacts and Procedures (4.2.2)
* Support for Evaluating Impacts of Procedure Execution
Functional Capability Assessment (4.2.2)
* Support for Determining Remaining Functionality after Failure
Unanticipated Situations and Workaround (4.2.3)
Support for Developing Workaround Procedures
Minimizing Risk Introduced by Workaround Procedures

Responding to anomalies in space systems is a complex decision-making process, often with high stakes (e.g., potential loss of crew, failure of costly experiment). The operator must make the following decisions when an anomaly occurs:

- Determine if any action is required in response to anomaly
- Distinguish failures from false alarms
- If more than one response is possible, select one
- When action is taken, evaluate if it is effective

It is important for systems to provide the right information to support the operator in making these decisions.

What tasks are performed in response to anomalies in the monitored process? After safing the vehicle, the first task is to identify the cause of the anomaly (or a set of possible causes). Next, the impacts of the anomaly are determined, including the immediate consequences to mission objectives and the potential for future consequences to both safety and mission objectives. Finally, response options are delineated and the "best" response is enacted. The execution of response procedures must be monitored to verify that they have the desired effect and to detect adverse side-effects. In the remainder of this section, we describe the decisions made by the space operations support team when performing these tasks and the information they need to respond effectively.

When an alarm is annunciated, the operator initially determines if a critical failure has occurred (i.e., an emergency). In an emergency, immediate action is taken to safe the crew and vehicle. If not an emergency, the operator proceeds to identify a cause for the anomaly. Further diagnosis is made to eliminate false alarms and to resolve redundant alarms. To interpret an alarm, the operator must understand the situation leading up to the alarm (e.g., relevant events and activities). This includes any changes to the normal state of the monitored process that have preceded this situation and that could affect the situation (e.g., changes to system configuration, state transitions, and existing failure conditions). Thus, understanding of situation is developed by providing the operator with visibility into how the monitored process has been behaving. Providing such visibility is necessary in

building continuously useful intelligent systems (i.e., systems that the operator relies upon, even during nominal operations; see also section 2.2). Such visibility is affected by what information is available to the operator and how much confidence the operator has in it (i.e., quality of information and believability of the source of information, including the source of alarms). When dealing with multiple alarms, the timing and sequence in which they were issued is useful in understanding how the situation developed (e.g., power fail causes loss of fan and overheating of device).

The immediate consequences of an anomaly are manifested as loss of functional capability. Functionality can be lost in both the monitored process and related systems. The impact of functional loss is determined by the criticality of the loss to safety (i.e., does the loss threaten crew or vehicle safety?) and operations (i.e., does the loss significantly[9] alter ongoing or future operations). Redundant capability can reduce the criticality of the loss by providing alternate ways to achieve mission goals. The ability to recover the lost functionality (e.g, repair) also reduces the criticality of the loss.

The consequences of taking no action are predicted to evaluate how the anomaly might affect other systems (or parts of systems). This assessment is called the *failure propagation potential* and is based on the functional dependencies among systems. The goal of this prediction is to identify when (and under what circumstances) unacceptable impacts could occur. Unacceptable impacts for space operations include violation of flight rules and irreversible or adverse effects on operations[10]. Propagation potential is used in evaluating anomaly response options. If the potential effects are severe or if the effects will happen soon, more drastic response options may be needed. This assessment also defines the time span for adverse effects to develop, thus bounding the available response time and setting expectations for the next likely failure.

The consequences of taking action must also be evaluated (i.e., determine the response options). This includes determining the available response strategies and evaluating how these strategies would impact safety and operations. Similar to the assessment of failure propagation potential, the effects of all options being considered must be predicted and evaluated for unacceptable impacts (e.g., don't have time to recover the lost system). A strategy is selected to minimize the risks associated with anomaly response (the discussion of reliable intelligent systems in section 2.2 is relevant to response options including the IS). There are two types of response options - *repair* or recovery of the anomalous system and *compensation* for lost capability. Compensation options include using alternative capability or modifying mission plans and objectives in light of the remaining capability.

In cases of contingency, response options may have to be developed in real time. Since contingencies are unfamiliar situations, the problem must first be understood before a response can be determined. Workaround procedures are often formed in response to contingencies by modifying planned procedures. Like all response options, these procedures should be evaluated for compliance with flight rules, risks, and negative impacts (e.g., changing configuration can complicate fault management by overwriting needed data). Note that contingency response can include reallocation of intelligent system tasks. As described in section 3.2, it is necessary to design the intelligent system for such situations (e.g., permit the human to inform the intelligent system of new procedures, or takeover responsibility from the intelligent system).

---

[9] Slight changes to operations are distinguished from severe changes.

[10] These effects are only considered adverse if they impact planned operations or safety (i.e., if failed system is not planned for use, no impact).

We have described the decisions that must be made by the space operations support team when responding to an anomaly. A significant amount of information must be provided to the team to support this decision-making. Common practice in presenting information for managing the monitored process (i.e., message lists and schematics) do not represent the right information (e.g., schematics illustrate failed components, not lost functionality). Methods for designing effective representations and presentations of this critical diagnostic information remains a research issue. A related issue is the need to manage information overload.

3.5 Information Requirements for Intelligent System Management

Possibly the most significant problem in intelligent system design is failure to recognize that an intelligent system poses new tasks for the operator (cf. also section 2.3). In addition to managing the monitored process, the operator must now manage the IS, including monitoring and coordinating its activities, and responding to its errors. But intelligent system are rarely designed to be managed. The traditional means of communicating with the IS (message lists, schematics, and explanation) do not support the operator in monitoring IS activities and understanding IS reasoning strategies. And, when IS errors occur, the operator has few options for responding to them (usually the restart button or the power plug).

This omission in intelligent system design arises from misconceptions about how intelligent systems are used in space operations. In reality, the intelligent system is embedded in a larger support system. Intelligent system activities need to be coordinated with the operator's other activities, and information from the IS needs to be integrated with other displays and data sources. Data are often noisy or unavailable. In all likelihood, the IS designer will have to provide some capability to handle data problems. Even with this precaution, unanticipated situations will occur. Then, the intelligent system will make errors. The operator, who is a highly trained expert and more knowledgeable than the IS, is responsible to detect and respond to IS errors.

This situation requires that the intelligent system be designed to support such response, and that it provide adequate information for the operator to understand what it is doing and know how to respond. Instead of an explanatory tutorial, the operator needs support in understanding situations as they develop, including the activities and strategies of the IS and the critical information characterizing the situation (e.g., evidence of fault in data). There is a risk of overloading the operator if the IS is not designed to assist the operator in performing these new tasks and managing the new information needed for these tasks. See section 2.1 for more information about how intelligent systems are used in space operations.

In section 3.5 we discuss the information that must be exchanged to manage the intelligent system. In section 3.5.1, we describe how the human communicates and collaborates with the IS to develop a shared view of situation that includes IS activities. In section 3.5.2, we describe how the human guides and corrects the IS in response to errors.

3.5.1 Communicating with the Intelligent System

Reference to the Tech Memo

Related Section

- 4.1.2 Collaboration between Agents
- 4.3 Workspace Design
- 5.3 Message Lists and Timeline Displays

Problems

Visibility into Intelligent System Activities and Reasoning (4.1.2)
- Providing Visibility into Intelligent System Reasoning
- Distinguishing Hypotheses from Facts
- Understanding Intelligent System Reasoning Strategy

Explanation (4.1.2)
- Responding to Questions in Context
- Making Abstract Entities Concrete
- * Promoting Shared Understanding of Situation
- * Improving Explanation for Agent Collaboration

Review (4.1.2)
- Providing Access to Event History
- Recording Information for Review

To manage the intelligent system, the operator must be able to exchange information with the intelligent system. Such communication is necessary for coordinating the activities of the space operations support team (see section 3.2 for a discussion of coordination) and for guiding and correcting the IS (see section 3.5.2 for a discussion on responding to IS errors). Yet the traditional approaches to IS communication are inadequate (i.e., schematics, message lists, explanation) because they don't represent the right type of information and they don't present it in an effective way (see also section 2.2). The goal of communication is to establish a shared understanding of events or situations affecting both the operator and the IS. This includes understanding what the IS is capable of doing (its capabilities), and what it is currently doing (its ongoing activities). We call this having some visibility into the intelligent system.

Providing visibility is providing an unobstructed view to the operator. To be *unobstructed*, nothing should get in the way of sight (i.e., information is clearly presented, with the important messages highlighted and the meaning obvious). A *view* includes a perspective. The perspective of the operator must be supported by providing information in terms that are expected and easily understood. Information for visibility into the monitored process is represented from the perspective of managing failures in the monitored process (see section 3.4.2). Information for visibility into the intelligent system is represented from the perspective of coordinating with and managing errors in the IS. Different information is needed for each perspective. Providing visibility into the intelligent system means clarifying IS conclusions and reasoning strategies[11] as a situation develops (i.e., what the IS concludes is going on in the monitored process, and what the IS is doing about it). It means showing IS activities in progress and how well these activities are achieving goals. It also means providing the operator with the information used to draw a conclusion (including critical evidence and operating assumptions). This information should be presented in a way that reinforces the operator's understanding of IS reasoning strategy

[11] Information about IS conclusions and reasoning strategies can include intermediate conclusions, interim states, hypotheses, and alternative solutions and strategies.

(Chandrasekaran *et al.*, 1989). Providing such visibility into the intelligent system is an important aspect of building a continuously useful system (see section 2.2).

Thus, managing the intelligent system means providing a lot of new information to the operator. To do this effectively, the designer must identify what information it is important to communicate and when to communicate it (i.e., the information exchange requirements, see section 2.4). Without such an analysis, the designer runs the risk of either providing insufficient information for the task (and risks developing a "magical" intelligent system, described in section 2.1), or overloading the operator with too much information. Other design issues related to providing visibility into the intelligent systems include displaying the confidence in information (distinguishing between hypotheses and conclusions), representation and presentation of IS reasoning strategy, and managing complexity of information presentation (e.g., use of overlays to manage complexity on large diagrams). See also the discussion of modes in section 3.2 as a means of implementing alternate IS strategies.

Explanation is the common approach to communicating with the intelligent system and is usually the only means of providing visibility into the intelligent system. We define *explanation* as the iterative process of identifying, organizing, and presenting information to satisfy the operator's inquiry about some situation or event. To understand the situation, the operator must understand both the environmental evidence and the information and reasoning strategies used by the IS used to draw conclusions about that evidence. Thus, explanation includes both asking for information about the monitored process, and for information about the IS decision-making process.

As described in section 2.2, explanation has some serious limitations for real-time operational use. Alternatives to explanation are needed that assist in developing a shared view of situation. Central to developing a shared view is the requirement that both the operator and the intelligent system monitor the same information. To come to the same conclusion, they must be looking at the same information. Thus, the operator doesn't have to understand the details of the IS reasoning mechanism, but must understand the bases of IS conclusions (i.e., critical environmental evidence; IS knowledge, activities, and strategies). To achieve this, it may be necessary to make implicit information explicit in the intelligent system (e.g., an explicit representation of robot tasks can enable the human to give task advice to a robot, Martin and Firby, 1991). Other considerations when designing a shared view include the following:

- Present information describing the situation as it develops, including both the situation with the monitored process, and the intelligent system activities associated with that situation.
- Establish expectations about what might happen next (can include reviewing information from predictions, or annotating operational data with predictions).
- Identify critical transitions and regimes of behavior (both current and pending; e.g., information about physical processes [and task] can be used to annotate displays, Forbus, 1991).

Notice that such information both supports the operator's tasks and provides some on-the-job training by reinforcing the operator's mental model of both the monitored process and the intelligent system.

A good first step towards developing a shared understanding of situation is to clarify the intelligent system reasoning strategies by showing the critical information used by the IS to

draw conclusions (i.e., the evidence supporting the conclusion). In the near term, we suggest that plots or tables showing a recent history of such "evidence" accompany the IS conclusions. Alternatives to messages lists, such as timelines (Potter and Woods, 1991), should also be considered for reviewing information (cf., section 5 of Tech Memo). In the long term, research into shared representations (e.g., Martin and Firby, 1991) may be a promising approach to designing alternatives to explanation.

There are some design issues associated with information review. Recording information for review can be memory intensive, thus imposing a design limitation on how review can be used. The designer must carefully identify what parameters to log, how often to log parameters, and when to log parameters to meet this limitation. It is also important to present information for review in a manner that distinguishes the ongoing situation from the past situation being reviewed. This is especially necessary if the same display formats are used for real-time monitoring and review.

3.5.2 Responding to Intelligent System Error

Reference to the Tech Memo

Related Section

4.1.3 Managing the Intelligent System
4.3.1 Interpretation of Information

Problems

Detection of Intelligent System Errors (4.1.3)
Identifying Intelligent System Errors
Intervention by Altering Information (4.1.3)
Altering Information to be Used by Intelligent System
Intervention into Reasoning Process (4.1.3)
* Redirecting Intelligent System Reasoning
Restart of the Intelligent System (4.1.3)
Support for Intelligent System Restart
Recording Input Data for Replay
Override of the Intelligent System (4.1.3)
* Support for Selective Override of Intelligent System Processing
* Support for Complete Override of Intelligent System
Risks of Intervention in Intelligent System Processing (4.1.3)
Avoiding Negative Impacts from Operator Intervention

The common approach to managing intelligent system error is preventive -- the software developer prevents intelligent system error from occurring by verification, validation, and testing to find the bugs. This approach is certainly necessary, but it is not sufficient for systems being used in complex and uncertain support environments like space operations. No matter how thorough the testing, the intelligent system can still make errors in unanticipated situations (i.e., situations that weren't tested). And, if not designed properly, there is nothing the operator can do to recover from the malfunction.

The concept of intelligent system error correction is usually expressed as designing an intelligent system that "gracefully degrades" near the boundaries of its knowledge or the limits of its performance. The notion of graceful degradation usually encompasses some limited form of self knowledge, in which the IS recognizes that it has gone beyond its limits and alerts the operator of the potential for error. We feel that graceful degradation can be achieved by providing the operator with the ability to take some action to

compensate or correct IS errors. In this section we discuss different approaches to designing intelligent systems that provide the operator with some options for responding to errors.

The first step in managing intelligent system errors is to understand how the IS can malfunction and to assist the operator in detecting malfunctions. Intelligent system errors can be caused by problems in the intelligent software (software bugs), by problems in input data (noisy or unavailable data), and by unanticipated behavior of the monitored process. As stated above, verification and validation techniques are the traditional way to eliminate software bugs. Data problems pose a different kind of design issue. The goal is to prevent (or at least minimize) the bad data processed by the intelligent system. This usually requires some data preprocessing or alarm management software to "filter out" the problematic data[12]. Although managing data problems may seem outside the scope of an intelligent software development project, such capability is necessary to build a reliable intelligent system. Since data preprocessing or alarm management software is usually not available elsewhere, the IS developer often has to develop such software as part of the intelligent system development project. See section 3.4.1 for a related discussion on false alarms and alarm management for the monitored process.

The intelligent system can malfunction in two ways: it can draw an incorrect conclusion or it can be unable to draw any conclusion. Incorrect conclusions result from data problems, or from an inaccurate or incomplete IS knowledge base. The inability to draw a conclusion can also indicate performance problems. Thus, to detect IS errors the operator must monitor both the accuracy of conclusions and the system's ability to produce conclusions in timely fashion. This requires that the operator understand the ongoing activities and the reasoning strategies of the IS (what we have defined as visibility into the intelligent system). For example, a sensor failure can effectively disable a portion of the knowledge base conditioned on data from the failed sensor. In such a case, the IS may be unable to reach a conclusion (i.e., cannot detect problems requiring data from the failed sensor). In effect, the IS will "just sit there", waiting for data that can never be received. The IS should provide intermediate information to illustrate what it is doing, and where its activities stopped. See section 3.5.1 for a related discussion on providing visibility into intelligent system activities and strategies.

The intelligent system should be designed to allow the operator to respond to its errors online. Most intelligent system are designed for off-line repair (i.e., when an error occurs, turn off the IS and debug it later). It is possible to provide some capability for on-line correction or compensation. The operator can correct or repair the malfunction by altering information used by the IS or by influencing the IS reasoning process. The operator can compensate for error by reallocating some or all of the intelligent system's responsibilities. It is also possible for the IS to include some self-correcting capability, particularly for errors resulting from bad data. For example, the Payload Deployment and Retrieval System (PDRS) Decision Support System (DESSY)[13] has self-correcting rules that retract erroneous assertions due to inconsistent data. It is likely that some combination of techniques from all of these approaches will be needed to build a reliable intelligent system.

The simplest form of intelligent system repair (and typically the only ability to intervene in IS processing) is the brute force approach of restarting the IS. Restarting the IS is a destructive approach, and is predicated on the assumption that the integrity of the IS

---

[12] Assessments of information source, quality, and validity can be used in detecting and managing information problems by indicating how trustworthy and reliable information is (see also section 4.3.1 in the Tech Memo).

[13] PDRS DESSY is being jointly developed by NASA/ER2 and Rockwell (Land *et al.*, 1992).

knowledge base is lost. Essentially, restarting the system is "starting anew". The knowledge base is restored, but at the cost of all the understanding of situation developed up to the time of the restart (i.e., reset all IS conclusions based on data). Restarting the IS from a checkpoint file (a file containing IS internal state information stored at a given time) can be used to minimize the information lost at a restart. Checkpoints can be made at operator request, periodically at some fixed rate, or event-triggered. Making checkpoint files, however, can be an additional processing load on the IS, and storing checkpoint files can be memory intensive.

A less common but more selective (and less destructive) approach to intelligent system repair is to intervene in IS processing. This can be accomplished by altering the information used by the IS or by modifying the IS reasoning strategy. Either data about the monitored process or information internal to the IS can be altered. For example, the operator can change a bad data value or fill in data unavailable from the monitored process. Alternately, the operator can modify an IS hypothesis or suggest an alternate solution (e.g, a new procedure). The operator can also alter the way information is processed by the IS, such as setting processing priorities of the IS (e.g., dynamically alter rule salience). If the IS has missed important data during the repair, or if the repair has inadvertently caused adverse or unexpected effects, it may be possible to recover by restarting the IS from a checkpoint taken prior to repair.

Reallocating intelligent system tasks is an alternative to repairing the intelligent system. The simplest form of reallocation is turning off the IS. Even this may be impossible, however, if the IS is not designed properly (i.e., when the IS is turned off, it may be impossible to access other needed information). Since IS software often provides the first opportunity to use graphics in the support environment, it is common to see a display upgrade for the entire support system as part of the IS development. In such cases, the designer must be especially careful not to make access to all data dependent upon access to the intelligent system.

It is possible to take a graded approach to task reallocation, by providing a range of possible allocations for the intelligent system from which the operator can select (what we call selectively overriding portions of IS responsibility). A common example of such selective override is providing operating modes that correspond to levels of responsibility (e.g, five levels of control over the IS [notification, confirmation, execution, partial takeover, complete takeover], Johns, 1990). Another way to provide selective override is to partition the knowledge base such that portions of it can be disabled by the operator (e.g., permit the operator to selectively disable rules). The ability to disable portions of the knowledge base can also be useful for handling errors due to performance problems. See the related discussion of alternate task allocations in section 3.2.

For some intelligent system errors, it may be necessary for the operator to completely override the IS (i.e., take over all IS responsibilities). To assume these responsibilities, the operator must be able to access critical operational information and control the monitored process independent of the IS. Operators should be trained in using the available override options, so these activities become familiar. Although a complete takeover effectively "turns off" the IS, it can be useful to continue running in background to log results for off-line debugging.

There are risks in intervening in intelligent system processing just as there are risks in intervening in the monitored process (see section 3.4.2). Unintentional or unexpected impacts of intervention can complicate or worsen the situation, and make it even more difficult to correct. Operator errors during intervention can also worsen the situation. If intervention requires the human to assume IS tasks, there is a risk that human will be

unable to do the task (or will make errors) due to insufficient understanding of situation, inadequate training, or overwork or distraction resulting from the additional task responsibilities. If intervention involves reconfiguring or commanding systems, the effects of those changes can be difficult to reverse.

The potential for making the situation worse means that the operator should be supported in exercising caution when intervening. The operator may need additional tools to evaluate the consequences of intervening into intelligent system processing. This support should assist the operator in understanding the situation and, as much as possible, evaluating the impacts of intervention prior to action. As a means of protecting against drastic impacts, it is possible to take a system checkpoint prior to intervention, so that the intelligent system can be returned to its former state should something go wrong. This technique is only partially effective, since the effects of intervention are not always reversible.

Section 4
Summary

This report provides an overview of the HCI design guidance for intelligent system designers documented in the NASA Technical Memorandum 104738 "Making Intelligent Systems Team Players" (Malin *et al.*, 1991). This overview report was written to summarize the important concepts contained in this Tech Memo, and to describe how this guidance can be used to design intelligent systems.

In section 2, we summarize the important, new concepts for intelligent system design. We characterize how space operations constrain intelligent system design. We clarify what is means for an intelligent system to be a team player, and describe the information requirements approach to designing team players. We define the important design problems and show how they affect intelligent system reliability and usability.

In section 3, we describe the HCI design guidance from the perspective of the intelligent system designer. We organize the recommendations into sections corresponding to the analysis and design activities described in the Tech Memo (see section 6.1). These activities produce information requirements for both the intelligent system and user interface design.

We have not discussed specific user interface designs in this report. User interface design examples and suggestions for information management are described in sections 4.3 and 5 of the Tech Memo. Particular topics addressed are workspace design and the use of schematics and message lists.

In this report, we identify many unresolved design issues requiring further investigation. These issues can be summarized as the following intelligent system design needs:

- Design Methods
  Design methods are needed that specify human and intelligent system interaction based on a task description. This task description should integrate intelligent system operations with other human tasks. The specification of human-intelligent system interaction should highlight changes in task allocation and tasks requiring close agent coordination (such as task sharing).

- Information Requirements
  New types of information are needed that support the operator in making important domain decisions (e.g., critical diagnostic information), and that assist the operator in supervising and coordinating with the intelligent system.

- Intelligent System Design
  New approaches to intelligent system design are needed that permit the operator to repair IS errors. Improved designs are also needed that accommodate the constraints of the space operations environment, including robustness to data problems and flexibility in unexpected situations.

- User Interface Design
  Alternatives are needed to common practice in user interface design for intelligent systems. These alternatives should support developing a shared human-intelligent system view of the situation. This requires new types of information, increasing the potential for information overload of the operator. New user interface designs are needed that integrate diverse sources of information by focusing on information relevant to the task. Designs are also needed for managing interruptions in multitasking situations.

In table 4-1, we show how these addressing these needs will solve the design problems affecting intelligent system reliability and usability (see table 2-1). We invite researchers from both the artificial intelligence and human factors communities to consider these needs in their research.

Table 4-1. Intelligent System Design Needs and Problems They Address

DESIGN PROBLEMS	IS DESIGN NEEDS
Problems Affecting Reliability	
1. IS not designed to support human operator in supervising IS activities and recovering from IS errors	• New design methods specifying user supervisory tasks as a team architecture with dynamic task allocation • Improved designs for repairable IS - Inform and guide IS, and assess impacts of intervention - Self-repair • Longer term, IS that "learn" from humans in real time
2. Unrealistic assumptions about space operations • bad or unavailable data • unexpected situations	• IS designs that are robust to data problems - Consider data "trustworthiness" (source, quality) - Operate effectively with incomplete information • IS designs that are flexible in unexpected situations - New information & capability for contingency response - Repairable IS (see 1 above)
3. IS activities, reasoning strategies, and capabilities misunderstood and often overestimated by human operator (i.e., a magical IS)	• New information and designs supporting shared human-intelligent system view - Monitor IS activities, and success of activities - Clarify IS reasoning and capabilities • Longer term, self-explanatory IS
Problems Affecting Usability	
1. Common practice in user interface (UI) design increases user workload and fails to provide the most usable information • message lists and schematics • explanation	• New types of information and UI designs supporting operator domain decisions (e.g., functional assessments) • New UI design methods for managing information overload by identifying task-relevant information • Alternatives to message lists and schematics that better support operator tasks (cf. Woods) • Alternatives to explanation based on shared understanding of situation
2. IS not integrated with support system	• New design methods that specify operation of IS as part of overall operator tasks • New UI designs that integrate diverse sources of information based on task
3. IS not designed for coordination	• New design methods that specify agent coordination, task sharing, and dynamic task allocation • New types of information supporting agent coordination and task sharing • New UI designs for managing interruption of operator in multitasking situations

References

Abbott, K. Nov 1991. Internal memo. Human Automation Integration Branch. Langley Research Center, NASA.

Abbott, K. 1992. Personal communication.

Andriole, S., and Adelman, L. 1991. Prospects for cognitive systems engineering. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, Charlottesville. VA.

Arkes, H.R., and K.R. Hammond. 1988. *Judgement and decision making: An interdisciplinary reader*. Cambridge University Press.

Bloom, Charles P. 1991. Discussion at *CHI '91 Workshop on Advances in Computer-Human Interaction in Complex Systems*, Human Factors in Computing Systems Conference, New Orleans, LA.

Card, S., Moran, T., and Newell, A. 1983. *The psychology of human-computer interaction*. Hillsdale, NJ: Erlbaum.

Chandrasekaran, B., Tanner, M., and Josephson, J. 1989. Explaining control strategies in problem solving. *IEEE Expert*, 4(1), 9-24.

Clark, C., Jowers, S., McNenny, R., Culbert, C., Kirby, S., and Lauritsen, J. 1992. Fault Management for the Space Station Freedom Control Center. In the *Proceedings of the American Institute of Aeronautics and Astronautics 30th Aerospace Sciences Meeting & Exhibit*, Reno, NV.

Decker, K. Garvey, A., Humphrey, M., and Lesser, V. 1991. Effects of parallelism on blackboard system scheduling. In *Proceedings of IJCAI*, Australia. 15-21.

Forbus, K. 1991. *Qualitative Physics as a Tool for Human-Computer Interaction*. The Institute for the Learning Sciences, Northwestern University. Evanston, IL.

Hill, T. 1991. *Thermal Control System Automation Project (TCSAP): Introduction and Overview*. Briefing on 11 December 1991, Houston, TX.: McDonnell Douglas.

Hollnagel, E. 1991. *A Goals-Means Task Analysis Method*. Draft, Version 1. Computer Resources International. Denmark.

Horvitz, E., Ruokangas, C., Srinivas, S., and Barry, M. 1992. Project Vista: Display of Information for Time-Critical Decisions. In *Proceedings of the Fourth Rockwell International Conference on Control and Signal Processing*, Anaheim, CA.

Johns, G. L. 1990. *Graphic Interfaces to Intelligent Fault Management Systems: Issues and Guidelines*. MTR-90W00103. The MITRE Corporation, Houston, TX.

Land, S. Culp, D., and Malin, J. 1992. A Decision Support System for RMS Flight Controllers. Presented at the Joint Applications in Instrumentation Process and Computer Control Symposium, Houston, TX.

Leveson, N. G. 1991. Software safety in embedded computer systems. *Communications of the ACM*, 34 (2), 34-46.

Malin, J., Schreckenghost, D., Woods, D., Potter, S., Johannesen, L., Holloway, M., and Forbus, K. 1991. *Making intelligent systems team players: Case studies and design issues. Vol.1: Human-Computer Interaction Design; Vol.2: Fault Management System Cases.* NASA Technical Report 104738, Johnson Space Center, Houston, TX.

Martin, C. E., and Firby, R. J. 1991. An integrated architecture for planning and learning. *ACM SIGART Bulletin*, 2(4), 125-129.

Potter, S. and Woods, D. 1991. Event driven timeline displays: beyond message lists in human-intelligent system interaction. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, Charlottesville, VA.

Sycara, K. P., and Lewis, C. M. 1991. Cooperation of heterogeneous agents through the formation of shared mental models. In *Proceedings of AAAI Workshop on Cooperation Among Heterogeneous Intelligent Agents*, Anaheim, CA.

Woods, D. D. 1991. Representation aiding: A ten year retrospective. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, Charlottesville. VA.

Woods, D., Potter, S., Johannesen, L., and Holloway, M. 1991. *Human Interaction with Intelligent Systems: Volume I -- Trends, Problems, New Directions.* CSEL Report 1991-001. Cognitive Systems Engineering Lab, Ohio State University, Columbus, OH.

Glossary

AI	artificial intelligence
DESSY	decision support system
ERF	extended real-time failure effects and analysis tool
HCI	human-computer interaction
IS	intelligent system
ISs	intelligent systems
MP	monitored process
PDRS	payload deployment and retrieval system
PRS	procedural reasoning system
RCS	reaction control system
RTDS	real time data system
SSCC	space station control center
TCSAP	thermal control system advanced automation project
UI	user interface

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1992	Technical Memorandum

4. TITLE AND SUBTITLE
Making Intelligent Systems Team Players: Overview for Designers

5. FUNDING NUMBERS

6. AUTHOR(S)
Jane T. Malin
Debra L. Schreckenghost*

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Lyndon B. Johnson Space Center
Automation and Robotics Division
Houston, Texas 77058

8. PERFORMING ORGANIZATION REPORT NUMBER
TM 104751

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, D.C. 20546

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
S-680

11. SUPPLEMENTARY NOTES
*The MITRE Corporation
Houston, Texas

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Subject Category 59

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report is a guide and companion to the NASA Technical Memorandum 104738, "Making Intelligent Systems Team Players," Volumes 1 and 2. The first two volumes of this Technical Memorandum provide comprehensive guidance to designers of intelligent systems for real-time fault management of space systems, with the objective of achieving more effective human interaction. This report provides an analysis of the material discussed in the Technical Memorandum. It clarifies what it means for an intelligent system to be a team player, and how such systems are designed. It identifies significant intelligent system design problems and their impacts on reliability and usability. Where common design practice is not effective in solving these problems, we make recommendations for these situations. In this report, we summarize the main points in the Technical Memorandum and identify where to look for further information.

14. SUBJECT TERMS
Human-computer interaction, user interface, intelligent system, design guidance, development methodolgy, real-time fault management

15. NUMBER OF PAGES
51

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited